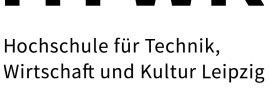
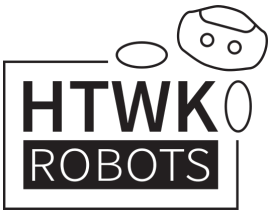




HTWK Leipzig
Fakultät Informatik und Medien
Studiengang Informatik

Bachelorarbeit

Training und Evaluation eines neuronalen Netzes zur Lösung der „Visual Referee Challenge“

Vorgelegt von: Freijdis Jurkat
geboren am: [redacted]
in:

Erstprüfer: Prof. Dr. Jens Wagner
Zweitprüfer: M.Sc. Tobias Kalbitz

Vorgelegt am: 15.05.2024

Danksagung

An dieser Stelle will ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt haben:

Mein besonderer Dank gilt Professor Jens Wagner. Ihre Begeisterung für Hardware und Robotik hat auch meine Faszination für dieses Feld entfacht, wodurch diese Arbeit erst zustande kommen konnte. Ihre kontinuierliche Motivation und Ihr unerschütterlicher Glaube an meine Fähigkeiten haben mir während meines gesamten Studiums immens den Rücken gestärkt.
Ebenfalls danke ich Tobias Kalbitz für die außerordentliche Betreuung. Deine wertvollen Ratschläge und Hilfestellungen haben mir geholfen, mich auf Kurs zu halten und die regelmäßigen Treffen waren von unschätzbarem Wert für meinen Fortschritt.
Des Weiteren möchte dem Dutch Nao Team für die Bereitstellung ihres Datensatzes und den Naos Hubble, Phoenix und Pioneer für die Mitwirkung bei den Aufnahmen meiner Posenvideos danken. Eure Mitarbeit hat es mir ermöglicht, die benötigten Daten für meine Arbeit zu sammeln.
Ein großer Dank geht auch an meine Mama und Solveig, für euren Blick von außen auf diese Arbeit während des Korrekturlesens. Eure Perspektive und Unterstützung haben mir geholfen, die Balance zwischen Fachsprache und Verständlichkeit zu finden.
Mein letzter Dank gilt Laurin. Du standest während meines gesamten Studiums hinter mir und hast mir immer wieder Mut gemacht, wenn ich selbst nicht mehr an mich geglaubt habe. Danke!

Kurzzusammenfassung

Die Schätzung von Posen ist ein bedeutendes Forschungsgebiet im Bereich der künstlichen Intelligenz, das die Mensch-Maschine-Interaktion vorantreibt und auch im Sport immer mehr an Relevanz gewinnt. Während menschliche Fußballspieler auf dem Feld mit den Schiedsrichtern ganz natürlich interagieren, wurde dieser Aspekt jedoch bisher in der Standard Platform League des Robocups vernachlässigt. Diese Arbeit untersucht einen weiteren Ansatz, um die Klassifizierung von statischen und dynamischen Schiedsrichterposen durchzuführen und damit dem großen Ziel, dass bis Mitte des 21. Jahrhunderts ein vollständig autonomes Roboter-Team nach den offiziellen FIFA-Regeln gegen den aktuellen Weltmeister gewinnen soll, einen Schritt näher zu kommen. Hierfür wurden Videos von relevanten Schiedsrichterposen erstellt und gesammelt. Anschließend wurden die menschlichen Gelenke mittels MoveNet extrahiert und die Pose mithilfe eines Convolutional Neural Networks klassifiziert. Dabei wurden zwei verschiedene Ansätze verfolgt: Ein Modell für jede Pose und ein Modell für alle Posen. Die Untersuchung zeigt, dass gute bis sehr gute Ergebnisse für statische und dynamische Posen erzielt werden können, wobei die Genauigkeit von einem Modell pro Pose 91,3% bis 99,3% mit einem Durchschnitt von 96,1% erreicht und die Genauigkeit von einem Modell für alle Posen eine Genauigkeit von 90,9% erreicht. Die erfolgreiche Anwendung der entwickelten Methodik zur Schätzung von Posen im Roboterfußball eröffnet vielversprechende Perspektiven für die Zukunft dieses Bereichs. Die gewonnenen Erkenntnisse können nicht nur zur Verbesserung der Leistungsfähigkeit von Fußballrobotern beitragen, sondern auch einen bedeutenden Beitrag zur weiteren Integration von KI-Technologien in unsere Gesellschaft leisten.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ANN Artificial Neural Network

CNN Convolutional Neural Network

CSV Comma-separated values

flops Floating Point Operations

GRU Gated Recurrent Unit

HPE Human Pose Estimation

k-NN k-Nearest Neighbors

KI Künstliche Intelligenz

LSTM Long Short-Term Memory

MMI Mensch-Maschine-Interaktion

MRI Mensch-Roboter-Interaktion

ReLU Rectified Linear Unit

RGB Rot, Grün, Blau

RNN Recurrent Neural Network

SPL Standard Platform League

1 Einleitung

Die Mensch-Maschine-Interaktion (MMI) erlangt zunehmend an Bedeutung, da sie einen entscheidenden Einfluss auf verschiedenste Aspekte unseres Lebens ausübt. Deswegen hat dieses Forschungsgebiet in den letzten Jahren einen signifikanten Aufschwung erfahren. Dies ist nicht zuletzt auf den verstärkten Einsatz von Robotern sowohl im alltäglichen Umfeld [1] als auch in industriellen Bereichen [2] zurückzuführen. Dieser Trend unterstreicht die Notwendigkeit eines umfassenden Verständnisses der Interaktion zwischen Mensch und Maschine. Ein Teil der MMI ist die Mensch-Roboter-Interaktion (MRI). Roboter agieren in immer engerer Kooperation mit Menschen und werden zunehmend als Kollegen betrachtet [3]. Diese Interaktion kann verschiedene Formen annehmen: Ein Mensch kann den Roboter steuern, als Mechaniker mit ihm interagieren oder sogar als echter Teampartner [4] zusammenarbeiten.
Der *RoboCup*, ein Wettbewerb, bei dem neben anderen Disziplinen Nao-Roboter (Abbildung 2.1) in Teams 7 vs. 7 im Fußballspiel gegeneinander antreten [5], bietet ein faszinierendes Zusammenspiel von Robotik, künstlicher Intelligenz und sportlichem Wettbewerb. Er präsentiert damit ein reichhaltiges Einsatzszenario, das der Forschung vielfältige Herausforderungen bietet. Dabei ist angestrebt, dass bis zur Mitte des 21. Jahrhunderts, eine vollständig autonome Mannschaft humanoider Roboter gemäß den offiziellen FIFA-Regeln ein Fußballspiel gegen den amtierenden Weltmeister gewinnt [6]. Das Setzen eines so ambitionierten Ziels erfordert eine Reihe von technologischen Durchbrüchen, um die gestellte Aufgabe zu bewältigen. Einen Teil der Roboterfußballspiele des RoboCups stellt die *Visual Referee Challenge* dar. In dieser Herausforderung geht es darum, dass die Naos, visuelle Signale in Form von Posen[1] von Schiedsrichtern erkennen. Diese Fähigkeit stellt einen entscheidenden Aspekt der Kontexterkennung und Vision in der MRI dar.
In dieser Arbeit widme ich mich der Lösung der Visual Referee Challenge. Dabei existiert eine zentrale Fragestellung: Wie können Roboter in Echtzeit menschenähnliche Posen erkennen und interpretieren? Die vorliegende Arbeit verfolgt das Ziel, einen Proof of Concept[2] für die konsistente und genaue Erkennung sowohl statischer als auch dynamischer Posen im Roboterfußball zu liefern. Der Fokus liegt dabei darauf zu ermitteln, ob Convolutional Neural Networks (CNNs) nicht nur statische, sondern auch dynamische Posen zuverlässig erkennen können.
In dieser Arbeit angestrebt ist die korrekte Interpretation von Bewegungen und Posen, da sie von entscheidender Bedeutung für den Roboterfußball ist. Nur so können Roboter zukünftig effektiv am Spiel teilnehmen und strategisch agieren. Doch darüber hinaus bietet diese Arbeit das Potenzial, die Interaktion zwischen Mensch und Maschine generell zu verbessern. Autonome Systeme können von zuverlässiger Posenerkennung profitieren, sei es in der Robotik, Assistenzsystemen oder der MRI.
Um die Forschungsfrage zu beantworten nutze ich ein zweistufiges Vorgehen. Zuerst habe ich einen Datensatz von Schiedsrichterposen gesammelt und nutze das *MoveNet*-Modell, um die einzelnen Gelenke des Schiedsrichters als Schlüsselpunkte zu erkennen. Diese Schlüsselpunkte dienen dann als Grundlage für die weitere Analyse. Ich nutze CNNs, die sich bereits in anderen Kontexten auf dem Nao bewährt haben, um die Posen aus den Schlüsselpunktdaten zu erkennen. Dabei entwickle und vergleiche ich zwei Ansätze: Binäre Klassifizierung und Multiklassenklassifizierung. Bei der binären Klassifizierung werden mehrere Modelle trainiert und jedes einzelne Modell auf die Erkennung einer bestimmten Pose spezialisiert. Bei der Multiklassenklassifizierung kann ein trainiertes Modell alle Posen gleichzeitig identifizieren.
In meiner Arbeit werde ich zunächst den RoboCup und das Einsatzszenario genauer beleuchten und in die Grundlagen neuronaler Netze einführen, um das Verständnis für mein Vorgehen zu erleichtern. Anschließend gebe ich einen Einblick in den aktuellen Stand der Forschung zur Posenerkennung. Die eigene Implementierung der bereits beschriebenen Ansätze sowie ein Vergleich dieser werden die zentralen Schwerpunkte meiner Arbeit bilden. Abschließend werde ich in einem Fazit auf die Vor- und Nachteile der verschiedenen Ansätze eingehen und einen Ausblick auf die zukünftige Handhabung und Erfüllung dieser Herausforderung im RoboCup geben.

[1] Pose bedeutet in diesem Kontext die Position der Hände und Arme im Verhältnis zum Rest des Körpers.
[2] Da es in dieser Arbeit um einen Proof of Concept geht, ist die Implementierung und Evaluation meiner Ansätze auf dem Nao nicht Ziel dieser Arbeit.

2 Einsatzszenario

2.1 Der RoboCup

Der RoboCup ist eine wissenschaftliche Initiative mit dem Ziel, den Stand der Technik voranzubringen und die Robotik- und Künstliche Intelligenz (KI)-Forschung zu fördern. Hierzu wurde ein anspruchsvolles und langfristiges Ziel definiert, das zudem eine öffentlich ansprechende Herausforderung darstellen sollte: Roboter sollten in der Lage sein, gegeneinander Fußball zu spielen [5]. Optimistisch ausgedrückt: „Bis Mitte des 21. Jahrhunderts soll eine Mannschaft aus völlig autonomen humanoiden Roboterfußballern ein Fußballspiel nach den offiziellen Regeln der FIFA gegen den Sieger der letzten Weltmeisterschaft gewinnen." [6] Die Regeln werden jährlich angepasst, um den Fortschritt zu fördern und die Leistung der Liga zu verbessern. [7]

2.2 Die Standard Platform League

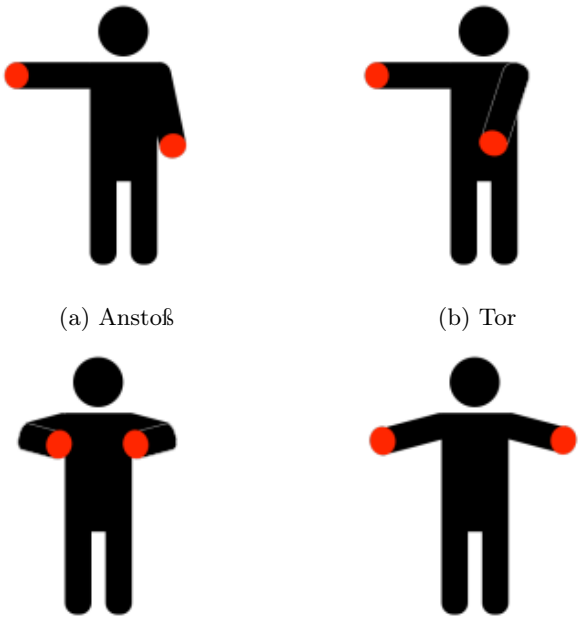
In der Standard Platform League (SPL) treten die Teams alle mit dem identischem Roboter 7 vs. 7 gegeneinander an, dem humanoiden Nao von Aldebaran (Abbildung 2.1). Die Roboter agieren auf dem Spielfeld völlig autonom und die Kommunikation nach außen findet nur über den *GameController* statt, der dazu da ist, einen fairen Wettbewerb zu gewährleisten, die Spielumgebung zu kontrollieren und die Roboter-Kommunikation zu überwachen, um die Integrität des Wettbewerbs zu sichern. Es sind keine Hardwaremodifikationen oder -erweiterungen erlaubt und somit ist die Performance der Roboter alleine von der Software des Teams abhängig. [8]
Neben dem Hauptwettbewerb existieren hier noch mehrere *Technical Challenges*, in denen die Teams Punkte sammeln können. Diese Challenges dienen dazu, um den Teams Zeit zu geben, technische Fähigkeiten zu entwickeln, die in den kommenden RoboCups im Hauptwettbewerb erwartet werden. 2023 waren das die *Dynamic Ball Handling Challenge*, in der nach zwei oder mehr schnellen Pässen ein Tor erzielt werden soll, die *Data Minimization Challenge*, in der es darum geht, die Datenübertragung zwischen den Robotern zu optimieren und die *In-Game Visual Referee Challenge*, in der die Naos die Aufgabe haben, Schiedsrichtergesten zu erkennen. [9]

Abbildung 2.1: Naos der HTWK Leipzig [10]

2.3 Die In-Game Visual Referee Challenge

Bislang muss der Roboter nur bei Anpfiff und dem Pfiff für ein Tor direkt auf den menschlichen Schiedsrichter reagieren. Sonst werden alle Entscheidungen und Statusupdates über den GameController kommuniziert. Hinsichtlich des langfristigen Ziels, dass ein Team komplett autonomer Roboter gegen den aktuellen Weltmeister im Fußball gewinnen soll [6], müssen die Roboter nicht nur auf die Pfiffe, sondern auch auf andere menschliche Signale reagieren und diese interpretieren können. Zu diesen Signalen gehören neben verbaler Kommunikation und dem Zeigen von Gegenständen (z.B. einer roten Karte) auch menschliche Posen. In der In-Game Visual Referee Challenge werden Posen in drei Kategorien abgedeckt: statische Zeichen mit einer Hand (Abbildung 2.2a), statische Zeichen mit zwei Händen (Abbildung 2.2b) und dynamische Zeichen mit zwei Händen (Abbildung 2.2c). [9]
Sollte sich ein Team dazu entscheiden, an der Challenge teilzunehmen, wird diese während aller Vorrundenspiele des Hauptwettbewerbes durchgeführt. Ein Schiedsrichter, gekleidet in Schiedsrichterkleidung

und roten Handschuhen, ist dafür veranwortlich, eines der 13 verschiedenen Zeichen zu performen, sobald ein Pfiff für Anstoß, Tor oder Ende einer Halbzeit erfolgt ist. Das Team hat dann während des normalen Spielablaufes 15 Sekunden Zeit, das Zeichen zu erkennen und ihre Bewertung an den GameController zu senden. Die Challenge beeinflusst das normale Spielgeschehen nicht. [9]
Mehr Informationen über den genauen Ablauf der Challenge und die Bedeutung und Beschreibung aller zu erkennenden Posen sind dem Anhang A.1 zu entnehmen.
Diese Arbeit beschäftigt sich ausschließlich mit dem Erkennen der Posen und nicht mit der Einleitung der Challenge durch einen Pfiff oder dem Senden der Bewertung an den GameController, daher wird im Folgenden nur noch von der *Visual Referee Challenge* gesprochen.



(a) Anstoß (b) Tor

(c) Spielende

Abbildung 2.2: Beispielposen für statische Pose mit einer Hand (2.2a), statische Pose mit zwei Händen (2.2b) und dynamische Pose mit zwei Händen (2.2c) [9]

3 Grundlagen neuronaler Netze

3.1 Artificial Neural Networks

Künstliche neuronale Netze (ANNs), sind Berechnungsmodelle, die sich an der Funktionsweise biologischer Nervensysteme, wie dem menschlichen Gehirn, orientieren. Solche Netze bestehen aus zahlreichen Knoten, auch als Neuronen bezeichnet, die miteinander verbunden sind und Signale austauschen können, ähnlich den Synapsen im Gehirn. Ein Neuron empfängt, wie in Abbildung 3.1 zu sehen, Informationen von Sensoren oder anderen Neuronen als Eingabe. Diese Eingaben werden durch einen Vektor von Gewichten modifiziert, der die Rolle von Synapsen zwischen den Neuronen nachahmt. Die Gewichtungen beeinflussen die Eingaben und werden zusammengeführt, um eine finale Eingabe zu erzeugen. Jedes Neuron in diesem Netzwerk verfügt über einen Schwellenwert. Es feuert oder gibt eine Ausgabe nur dann ab, wenn die Nettoeingabe diesen Schwellenwert, im folgenden Bias genannt, erreicht oder überschreitet. [11][12]

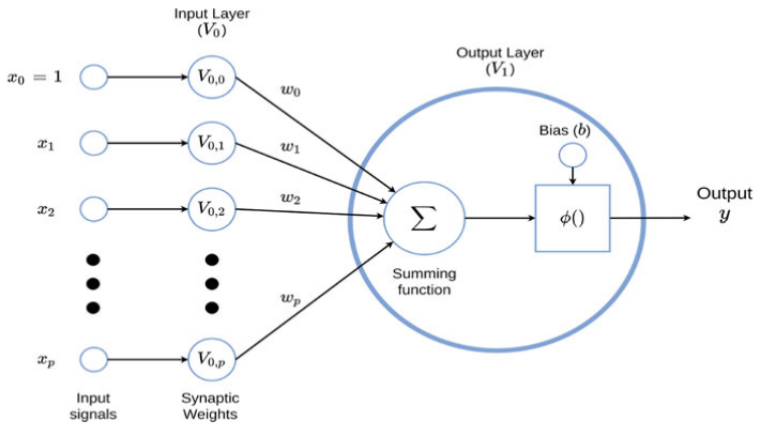


Abbildung 3.1: Generelles Modell eines ANNs mit Eingabeinformationen x_1 bis x_p, Inputlayerneuronen $V_{0,1}$ bis $V_{0,p}$, Gewichtsvektor $w = (w_0, ..., w_p)$, Outputlayerneuron V_1 mit Summenfunktion und Bias b und Output y [11]

Ein ANN kann beliebig viele Schichten von Knoten enthalten. Jeder Knoten in einer Schicht ist mit jedem Knoten in der folgenden Schicht verbunden, wodurch eine umfassende Verbindung zwischen den Schichten entsteht. [11][12]
Ein Beispiel für ein neuronales Netz mit vollständig verbundenen Schichten ist in Abbildung 3.2 zu sehen.

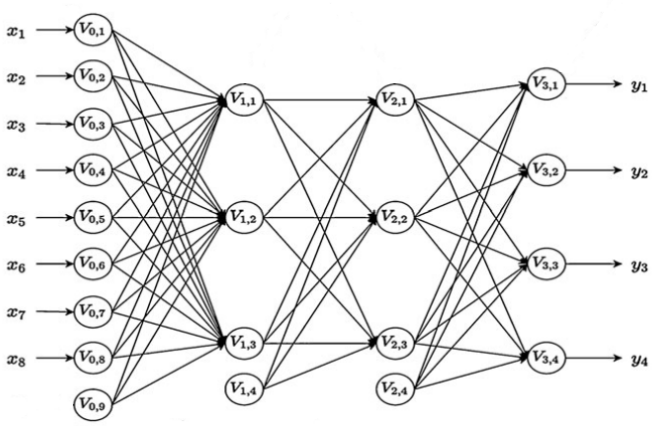


Abbildung 3.2: ANN mit einer Eingabeschicht V_0 mit acht Neuronen, zwei versteckten Schichten V_1 und V_2 mit jeweils drei Neuronen und einer Ausgabeschicht V_3 mit vier Neuronen. Die Neuronen $V_{0,9}$, $V_{1,4}$ und $V_{2,4}$ dienen als Biasneuronen. In Anlehnung an [11]

3.2 Convolutional Neural Networks

CNNs stellen einen speziellen Fall von ANNs dar, der sich besonders für maschinelles Lernen mit Bilddaten und bildgesteuerte Mustererkennungsaufgaben eignet. Bei klassischen ANNs führt das Training auf Eingaben wie Bildern zu Modellen, die zu groß sind, um effektiv trainiert zu werden. Dafür verantwortlich ist die vollständig verbundene Art der ANN-Neuronen. Um dem entgegenzuwirken, wird bei einem CNN ein Neuron eines Layers nicht mit dem kompletten vorderen Layer verbunden, sondern durch einen sogenannten Filter oder Kernel nur mit einer kleinen Region dieses Layers, was dazu beiträgt, die Anzahl der Parameter im neuronalen Netzwerk erheblich zu reduzieren. Diese Reduktion ist entscheidend, um die Effizienz des Trainingsprozesses zu verbessern und die Verarbeitung großer Eingabemengen zu ermöglichen. [13][14]

3.2.1 Architektur

CNNs beinhalten neben dem Input und Output Layer typischerweise mehrere Arten von versteckten Schichten, die übereinandergestapelt werden: Convolutional Layer (Faltungsschicht), Pooling Layer, und Fully-Connected Layer. [13]

- Der **Input Layer** repräsentiert das Bild und besitzt die Dimensionen Höhe x Breite x Tiefe, wobei die Tiefe beispielsweise die Rot, Grün, Blau (RGB)-Werte des Bildes mit einem Wert von 3 darstellen kann. [13][14]
- Der **Convolutional Layer** spielt eine entscheidende Rolle bei der Merkmalsextraktion in einem CNN. Er bestimmt die Aktivität der Neuronen, die mit lokalen Bereichen der Eingabe verbunden sind, indem er das Skalarprodukt zwischen den Gewichten des Kernels und der Region der Eingabe berechnet. Jeder Kernel definiert unterschiedliche Gewichte und ist darauf ausgelegt, spezifische Merkmale zu erkennen. Das Konzept der namensgebenden Convolution besteht darin, mit jeder Iteration den Kernel über die Eingabe schrittweise zu verschieben, anstatt das gesamte Bild auf einmal zu betrachten. Dies ermöglicht eine effiziente Merkmalsextraktion und reduziert die Anzahl der benötigten Gewichte pro Kernel. Diese Methode wird auch als Parametersharing bezeichnet, da dieselben Gewichte für verschiedene räumliche Regionen verwendet werden. Durch Parametersharing werden Merkmale unabhängig von ihren räumlichen Eigenschaften erkannt. Das Vorgehen, verschiedene Convolutional Layers mit unterschiedlichen Kernels zu stapeln, ermöglicht die schichtweise Erkennung immer komplexerer Merkmale im Bild. Dieser hierarchische Ansatz trägt dazu bei, die Repräsentation von Merkmalen in einem neuronalen Netzwerk zu verbessern und schrittweise komplexere Muster in den Daten zu identifizieren. [13][14]

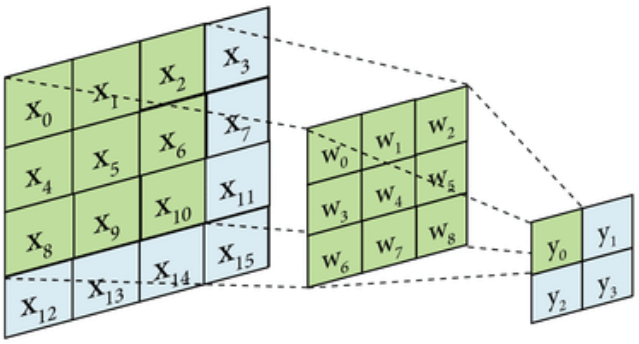


Abbildung 3.3: Das Neuron y_0 ist durch einen Kernel mit der Eingabeschicht x verbunden. Die Errechnung der Ausgabe y_0 erfolgt durch linerare Kombination von Eingabevektor $x = (x_0, x_1, x_2, x_4, x_5, x_6, x_8, x_9, x_{10})$ und Gewichtsvektor $w = (w_0, ..., w_8)$. In Anlehnung an [15]

- Der **Pooling Layer** ist für die Minimierung der Anzahl der Parameter und die dadurch reduzierte Rechenlast für nachfolgende Schichten verantwortlich. Der meistgenutzte Ansatz in CNNs dafür ist das Max Pooling. Dieser Vorgang teilt das Bild in Teilbereiche auf und gibt lediglich den maximalen Wert in jedem Bereich zurück, wodurch die Daten effizienter repräsentiert werden können. Neben dem Max Pooling gibt es auch noch andere Pooling Arten, wie zum Beispiel

das Average Pooling, bei dem für jede Region der Durchschnitt aller Werte berechnet wird. [13][14][16]

Abbildung 3.4: Max Pooling und Average Pooling mit Kernelgröße 3 und Stride 3 (siehe Abschnitt 3.2.3) [16]

- Der **Fully-Connected Layer** spiegelt die Struktur traditioneller neuronaler Netze wieder, bei der sämtliche Knoten mit allen Knoten in den angrenzenden Schichten verbunden sind. Seine Hauptaufgabe besteht darin, Klassifizierungsergebnisse aus den Aktivierungen zu generieren. Diese Schicht beansprucht die meisten Parameter innerhalb von CNNs und erfordert daher auch die längste Zeit und Rechenleistung während des Trainings. [13][14]
  Damit der Fully-Connected Layer die Ausgaben der vorherigen Schichten eines CNNs verarbeiten kann, müssen diese zuvor durch Flattening vereinfacht werden. Dabei werden die mehrdimensionalen Datenstrukturen in eindimensionale umgewandelt. [17]
- Der **Output Layer** nimmt die Ausgabe der vorherigen Schichten eines neuronalen Netzwerks und erstellt eine komprimierte Repräsentation, die die Wahrscheinlichkeiten oder Scores für verschiedene Klassen darstellt. [13]

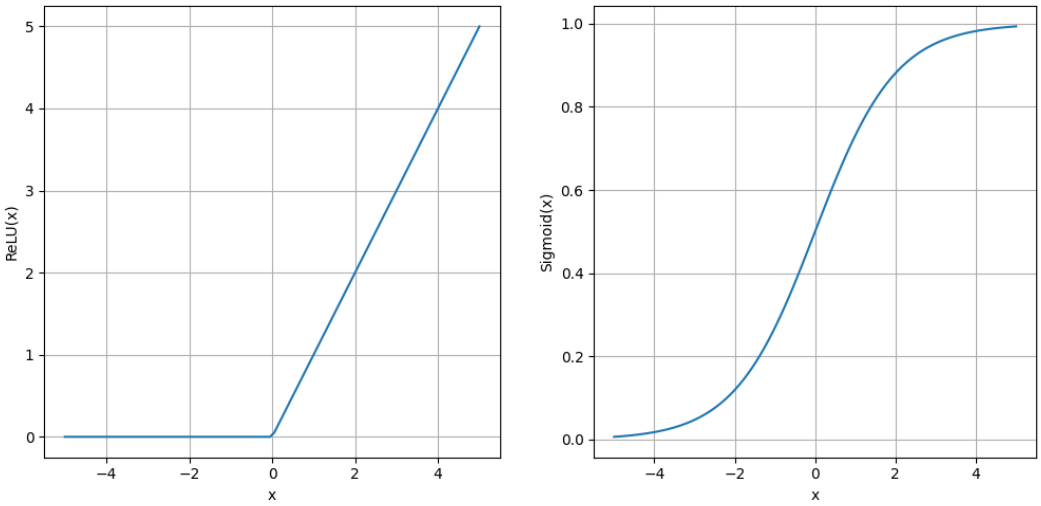
3.2.2 Aktivierungsfunktionen

Eine Aktivierungsfunktion wird auf jede einzelne Zahl der Ausgabe der linearen Kombination aus Eingabevektor und Gewichtsvektor der vorherigen Schicht eines neuronalen Netzwerks angewendet [13][14]. Sie transformiert ein Eingangssignal in das Ausgangssignal des nächsten Neurons. Ohne Aktivierungsfunktionen wäre ein neuronales Netz nur ein lineares Regressionsmodell, was nur begrenzt dazu in der Lage ist, Zuordnungen von komplizierten Datentypen zu erlernen und zu erkennen. [18]

- Die **Rectified Linear Unit (ReLU)** (Abbildung 3.5a) ist eine beliebte Aktivierungsfunktion, die in neuronalen Netzwerken in Convolutional Layers verwendet wird und deren Anwendung dazu führt, dass alle negativen Werte in der Aktivierung auf null gesetzt werden [13][14].
  Die ReLU-Funktion hat folgende Definition [14][18]:
  $$\text{ReLU}(x) = \max(0, x)$$
- Die **Sigmoid**-Funktion (Abbildung 3.5b) transformiert die Werte in den Bereich von 0 bis 1 und kann somit für die binäre Klassifzierung im Fully-Connected Layer verwendet werden. [18]
  Die Sigmoid-Funktion hat folgende Definition [18]:
  $$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

- Die **Softmax**-Funktion gibt wie die Sigmoid-Funktion Werte im Bereich von 0 bis 1 zurück. Sie berechnet die Wahrscheinlichkeiten für jede Klasse, die aufsummiert 1 ergeben. Die Softmax-Funktion wird deswegen im Fully-Connected Layer für Multiklassenklassifizierung angewendet. [18]
  Die Softmax-Funktion hat folgende Definition [18]:

$$\text{Softmax}(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \text{ für j} = 1, \ldots, \text{K}$$



(a) Aktivierungsfunktion ReLU (b) Aktivierungsfunktion Sigmoid

Abbildung 3.5: Visualisierung der Aktivierungsfunktionen ReLU und Sigmoid

Neben den eben beschriebenen Aktivierungtsfunktionen gibt es noch andere, jedoch ist die Verwendung von ReLU in den versteckten Schichten [18], Sigmoid für die binäre Klassifizierung [18] und Softmax für die Multiklassenklassifizierung [19] am geläufigsten.

3.2.3 Weitere Optimierungsmöglichkeiten

Neben der Variierung der Anzahl der verwendeten Knoten und Kernel und dem Einsatz von Pooling Layers gibt es noch weitere Möglichkeiten, um die Performance und Ausgabe eines CNNs zu optimieren:

- Die **Stride**, oder Schrittlänge, definiert, wie weit der Kernel mit jeder Iteration über die Eingabe verschoben wird. Eine größere Schrittlänge führt zu einer Verringerung der Ausgabegröße, während eine kleinere Schrittlänge zu einer größeren Ausgabegröße führt. [13][14]

- **Zero-Padding** ist ein einfacher Prozess, bei dem der Rand der Eingabe mit Nullen aufgefüllt wird. Dieser Vorgang ist äußerst effektiv, um möglichen Informationsverlust an den Rändern des Bildes zu verhindern. Durch das Hinzufügen von Nullen wird sichergestellt, dass der Kernel über die gesamte Eingabe gleiten kann, ohne wertvolle Informationen zu verlieren. [13][14]

- Die **Batch-Normalization** ist eine Technik, die dabei hilft, mit der Situation umzugehen, wenn sich die Verteilung der Daten in den verschiedenen Schichten des Netzwerks ändert. Diese Veränderungen können das Training verlangsamen und die Genauigkeit des Modells beeinträchtigen. Die Batch-Normalisierung funktioniert, indem sie die Daten in jeder Schicht des Netzwerks während des Trainings normalisiert. Dies hilft dabei, die Stabilität des Trainingsprozesses zu verbessern und ermöglicht es, höhere Lernraten zu verwenden, was zu einem schnelleren Training führt. [20]

- Die **Dropout**-Technik ist eine Methode, die effektiv Overfitting in neuronalen Netzwerken bekämpft. Zu Overfitting kommt es, wenn das Modell die Trainingsdaten mit der Zeit auswenig lernt und dadurch eine schlechte Leistung auf neuen Daten erzielt, also nicht gut generalisieren kann. Bei der Dropout-Technik werden zufällige Knoten und ihre Verbindungen vorübergehend

deaktiviert, um die Anpassung der Neuronen an die Trainingsdaten zu begrenzen. Diese vorübergehende Deaktivierung findet nur während des Trainings statt und nicht während des Testens oder der Klassifizierung durch das fertig trainierte Modell. Zusätzlich zur Verhinderung von Overfitting hat sich gezeigt, dass die Dropout-Technik durch das temporäre Eliminieren von zufälligen Parametern auch das Training beschleunigen kann [14]. [21]

3.3 Verschiedene Lernmethoden

Lernen bei Neuronalen Netzen bezeichnet die Methode der Modifikation der Gewichte und Biases eines Netzwerks. Während des Lernprozesses werden diese Parameter kontinuierlich angepasst, um die Ausgabe des Netzes zu optimieren. Dieser Anpassungsprozess trägt dazu bei, dass das neuronale Netzwerk sich an verschiedene Aufgaben anpassen und die gewünschten Ergebnisse liefern kann. [11][22]
Es gibt drei Lernmethoden: Supervised Learning, Unsupervised Learning und Reinforcement Learning.

- Beim **Supervised Learning** agiert eine externe Instanz oder ein Lehrer, die oder der das Training überwacht und anleitet. Dabei werden gelabelte Daten bereitgestellt, wobei die wahre Klasse der Daten bekannt ist. Das System nutzt diese Eingabe-Ausgabe-Paare, um die Eingabedaten in Ausgabedaten umzuwandeln. Während des Trainings betrachtet das neuronale Netzwerk die Abweichung zwischen den Ausgaben, die es produziert hat, und den tatsächlich erwarteten Ausgaben. Die Anpassung der Gewichte zielt darauf ab, die Ausgaben des Netzwerks so nah wie möglich an die gewünschten Ausgaben anzupassen. Wenn das Netzwerk das Ziel erreicht, wird die Übereinstimmung zwischen den gewünschten und den tatsächlichen Ausgaben maximiert. [22][23]
  Diese Methode eignet sich gut für Aufgaben, bei denen bereits gelabelte Trainingsdaten verfügbar sind. Typische Anwendungen sind Klassifizierungen, wie zum Beispiel die Vorhersage von Kreditrisiken oder Objekterkennung.

- Beim **Unsupervised Learning** agiert das System ohne die Anwesenheit eines Lehrers oder einer vorgegebenen Klassifizierung der Daten. Es versucht, Muster oder Strukturen in den Daten zu erkennen, ohne dass diese vorher explizit gelabelt wurden. Statt dessen analysiert das System die Daten, um verborgene Strukturen oder Zusammenhänge zwischen den verschiedenen Merkmalen zu finden. Während des Trainings muss das System versuchen, die Daten in sinnvolle Gruppen oder Kategorien zu ordnen, ohne dass ihm gesagt wird, welche Gruppen es bilden soll. [22][23]
  Unsupervised Learning eignet sich gut für Aufgaben wie Clustering und Anomalieerkennung, bei denen das Ziel darin besteht, Gruppen von ähnlichen Datenpunkten zu identifizieren oder Abweichungen von normalen Mustern zu finden.

- Beim **Reinforcement Learning** liegen keine präzisen Eingabe- und Ausgabedatensätze vor. Das System lernt durch die Interaktion mit seiner Umgebung oder Umwelt, welche auf das System mit positiven oder negativen Rückmeldungen reagiert, anstatt klare Anweisungen zu geben. Das System wählt seine nächsten Aktionen auf der Grundlage seiner früheren Informationen und auch durch neue Entscheidungen aus. Das Ziel ist es, die positive Rückmeldung zu maximieren, indem das System eine optimale Strategie entwickelt, um in seiner Umgebung zu handeln. [23][24]
  Reinforcement Learning ist gut geeignet für dynamische Entscheidungsprobleme, wie zum Beispiel Spiele oder die Steuerung von autonomen Fahrzeugen.

3.4 Evaluation

Eine Konfusionsmatrix (Tabelle 3.1) ist eine Darstellung, die verwendet wird, um die Leistung eines Modells zu bewerten. Sie gibt einen Überblick darüber, wie gut es die Daten klassifiziert, indem sie die tatsächlichen Klassen und die vorhergesagten Klassen darstellt. Die Konfusionsmatrix besteht aus einer Tabelle mit zwei Dimensionen: den tatsächlichen Klassen und den vorhergesagten Klassen. Jede Zeile repräsentiert die tatsächliche Klasse und jede Spalte die vorhergesagte Klasse. In der Konfusionsmatrix werden die folgenden Kennzahlen dargestellt: [25]

- True Positive (TP): Die Anzahl der Instanzen, die korrekt als positiv vorhergesagt wurden.

- False Positive (FP): Die Anzahl der Instanzen, die fälschlicherweise als positiv vorhergesagt wurden (während sie tatsächlich negativ waren).

- True Negative (TN): Die Anzahl der Instanzen, die korrekt als negativ vorhergesagt wurden.

- False Negative (FN): Die Anzahl der Instanzen, die fälschlicherweise als negativ vorhergesagt wurden (während sie tatsächlich positiv waren).

		True Class: negative	True Class: positive
Predicted Class	negative	True Negative (TN)	False Negative (FN)
	positive	False Positive (FP)	True Positive (TP)

Tabelle 3.1: Konfusionsmatrix für die binäre Klassifizierung

Aus diesen Kennzahlen können verschiedene Metriken abgeleitet werden, die genauer Auskunft über die Leistungsfähigkeit des Modells geben: Genauigkeit (Accuracy), Präzision (Precision), Rückruf (Recall) und F1-Score. Je höher die Kennzahl, desto besser ist das Modell. [25]

- Die **Genauigkeit** misst den Prozentsatz der korrekt klassifizierten Instanzen im Verhältnis zur Gesamtzahl der Instanzen. [25]

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- Die **Präzision** misst den Anteil der tatsächlich positiven Instanzen unter den Instanzen, die das Modell als positiv vorhergesagt hat. Eine hohe Präzision bedeutet, dass das Modell nur wenige falsch positive Vorhersagen macht. [25]

$$\text{Precision} = \frac{\text{TP} + \text{FP}}{\text{TP}}$$

- Der **Rückruf** misst den Anteil der tatsächlich positiven Instanzen, die das Modell korrekt als positiv vorhergesagt hat. Eine hohe Rückrufquote bedeutet, dass das Modell nur wenige falsch negative Vorhersagen macht. [25]

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- Der **F1-Score** ist das harmonische Mittel aus Präzision und Rückruf und bietet eine einzige Metrik zur Bewertung der Klassifizierungsleistung. Er kombiniert die Präzision und den Rückruf und ist besonders nützlich, wenn ein ausgewogenes Verhältnis zwischen Präzision und Rückruf erforderlich ist. [25]

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

4 State of the Art

Zur Lösung der Problems der Human Pose Estimation (HPE) gibt es mehrere Ansätze, allen gemein ist jedoch der zweistufige Prozess der Verarbeitung der Rohdaten: Zuerst erfolgt eine Erkennung verschiedener menschlicher Gelenke und Gesichtfeatures wie Nase, Augen und Ohren als Schlüsselpunkte mittels Gelenkerkennungsmodellen wie *MoveNet* [26], *PoseNet* [27], *BlazePose* [28], *OpenPose* [29] oder *Detectron2* [30]. Danach werden die erkannten Schlüsselpunkte mit unterschiedlichen Methoden interpretiert, um die Pose zu identifizieren. Diese zweite Stufe wird mittels Machine Learning oder Deep Learning Ansätzen umgesetzt.

4.1 Machine Learning Ansätze

Maschinelles Lernen ist ein Bereich der künstlichen Intelligenz, der sich mit der Entwicklung von Algorithmen und Modellen befasst, die es Computern ermöglichen, aus Daten zu lernen und Muster oder Zusammenhänge zu erkennen, ohne explizit darauf programmiert zu werden. Im Wesentlichen ermöglicht Maschinelles Lernen es Computern, automatisch aus Erfahrungen zu lernen und ihre Leistung bei bestimmten Aufgaben im Laufe der Zeit zu verbessern. [31]
Im Bereich der Posenerkennung spielen Decision Trees und der k-Nearest Neighbors (k-NN)-Algorithmus eine wichtige Rolle.

4.1.1 Decision Trees

In *An AI Approach to Pose-based Sports Activity Classification* werden drei Hauptklassen zur Identifikation von Tennisposen definiert: Vorhandbewegung, Rückhandbewegung und die Basisposition des Spielers. Zur Umsetzung dieses Ansatzes wird das Detectron2-Framework verwendet. Um die Pose zu klassifizieren, werden aus den erkannten Schlüsselpunkten für Tennis relevante, spielbezogene Merkmale abgeleitet. Dazu gehörten Winkel an Knien, Ellbogen, Hüften und Schultern sowie Abstände zwischen verschiedenen Körperpartien. Die Weiterverarbeitung erfolgt mittels des Decision-Tree-Modells Random Forest[3]. Jede Baumklasse gibt eine Klassenprognose aus und die Klasse mit den meisten Stimmen wird als endgültige Modellprognose gewählt. Bei statischen Testbildern erreicht das Modell eine Klassifizierungsgenauigkeit von 98,60%, während bei der Analyse von Videos eine Genauigkeit von 97,50% erzielt wird. [33]

4.1.2 k-NN Algorithmus

Die Arbeit *Yoga Pose Detection Using Posenet and k-NN* betrachtet statische Yoga-Posen wie Göttin, Berg, Planke, Baum und Krieger. Die Schlüsselpunkte werden mittels PoseNet bestimmt und anschließend mithilfe des k-NN-Algorithmus[4] klassifiziert. Das System erzielt bei Echtzeit-Tests eine durchschnittlichen Genauigkeit von 94%. [35]

Google schlägt den k-NN-Algorithmus für die Entwicklung eines Fitnesstrackers für das Zählen von Wiederholungen während des Trainings vor. Hierfür werden Bilder sowohl von der oberen als auch von der unteren Position einer Übung gesammelt und mithilfe von BlazePose werden die Schlüsselpunkte extrahiert. Anschließend werden diese Schlüsselpunkte in einen Featurevektor umgewandelt, der Merkmale wie den Abstand zwischen Handgelenk und Schulter, Knöchel und Hüfte sowie zwischen linkem und rechtem Handgelenk enthält. Um die Wiederholungen zu zählen, überwacht ein Algorithmus den Wahrscheinlichkeitsgrenzwert einer Zielposition. Sobald die Wahrscheinlichkeit der Posenklasse „nach unten“ einen bestimmten Grenzwert überschreitet, kennzeichnet der Algorithmus, dass diese Position eingenommen wurde. Sobald die Wahrscheinlichkeit unter den Grenzwert fällt, markiert der Algorithmus das Ende der Position und erhöht den Zähler für die Wiederholungen. [36]
Eine explizite Erfolgsquote ist hier nicht angegeben.

4.2 Deep Learning Ansätze

Deep Learning ist ein Teilbereich des maschinellen Lernens und bezieht sich auf künstliche neuronale Netzwerke. [37]
Bei der Posenerkennung wurden bereits Ansätze mittels ANNs, CNNs und Recurrent Neural Networks (RNNs)[5] verfolgt.

[3] Ein Random Forest ist ein Zusammenspiel aus vielen individuellen Entscheidungsbäumen. [32]
[4] Der k-NN-Algorithmus basiert auf dem Konzept, ähnliche Punkte in der Nähe voneinander zu finden. [34]
[5] RNNs sind Deep-Learning-Modelle, die durch interne Speicher sequenzielle Abhängigkeiten erfassen können, was sie für Aufgaben mit zeitlicher Abfolge geeignet macht. [38]

4.2.1 Artificial Neural Network

Die Arbeit *Yoga Pose Estimation and Feedback Generation Using Deep Learning* befasst sich mit der Anwendung von Deep Learning-Techniken zur Schätzung von Yoga-Posen. Dabei werden sechs statische Yoga-Posen betrachtet: Lotus, Kobra, Leiche, Berg, Dreieck und Baum. Einzelne Frames aus einem Video dienen als Eingabe für das OpenPose-Modell. Anschließend werden die Winkel zwischen den Schlüsselpunkten berechnet und an ein ANN zur Klassifizierung der Pose unter den sechs Yoga-Posen gesendet. Die vorgeschlagene Methode erreicht eine Genauigkeit von 99,58%. [39]

In *A Study of Human Fitness Pose Classification Using Artificial Neural Networks* werden 10 verschiedene dynamische Fitnessbewegungen klassifiziert. Mittels MoveNet werden die Positionen der Gelenke erkannt und einem ANN übergeben, was eine Genauigkeit etwa 85% erreichte. [40]

Auch Tensorflow schlägt die Verwendung eines ANNs zur Yoga-Posenklassifizierung vor. Dafür stellt Tensorflow einen Datensatz von Bildern mit den Yoga-Posen Stuhl, Kobra, herabschauender Hund, Krieger und Baum bereit. Nach der Erkennung der Schlüsselpunkte mittels MoveNet soll ein ANN trainiert werden, das die Yoga-Posen anhand der Gelenkpositionen erkennen kann. [41]
Eine explizite Erfolgsquote ist hier nicht angegeben.

4.2.2 Convolutionan Neural Network

Die Arbeit *BowlingDL: A Deep Learning-Based Bowling Players Pose Estimation and Classification* beschäftigt sich mit der Klassifizierung statischer Posen kurz vor dem Abwurf beim Bowling. Dabei werden fünf Kategorien unterschieden: Einhändig Links, Einhändig Rechts, Zweihändig Links, Zweihändig Rechts und Unkorrekt. Zur Extrahierung der Schlüsselpunkte wird das MoveNet-Modell verwendet. Die Weiterverarbeitung der Schlüsselpunkte erfolgt durch das eigens trainierte *BowlingDL*-Modell, das auf der Architektur eines CNNs basiert. Dieses Modell nimmt die Schlüsselpunktinformationen der Bowling-Spielerpose, die von MoveNet erkannt wurden, und klassifiziert sie in eine der fünf Kategorien. Die Methode erreicht eine Genauigkeit von 83%. [42]

Eine ähnlicher Ansatz wird in *Yoga pose classification: a CNN and MediaPipe inspired deep learning approach for real-world application* verfolgt. Hier jedoch werden die Schlüsselpunkte mittels BlazePose detektiert und mit dem eigenen CNN-Modell *YogaConvo2d* mit einer Genauigkeit von 99.62% in die Kategorien herabschauender Hund, Göttin, Planke, Baum und Krieger eingeordnet. [43]

Die Arbeit zum Thema *Pilates Pose Classification Using MediaPipe and Convolutional Neural Networks with Transfer Learning* beschäftigt sich mit der Klassifizierung von fünf Arten statischer Pilates-Posen (Herabschauender Hund, Göttin, Baum, Planke und Krieger). Als Merkmalsextraktor wird BlazePose verwendet. Die Ergebnisse werden anschließend von vortrainierten CNN-Architekturen mit Transferlernen[6] klassifiziert: *MobileNetV2*, *Xception* und *ResNet50*. Dabei wird die letzte Schicht der Modelle an die Anzahl der Pilates-Posenklassen angepasst, und die Gewichte der vortrainierten Modelle werden durch weiteres Training auf den neuen Datensatz angepasst, um die Leistung des Modells auf die spezifische Klassifizierungsaufgabe zu verbessern. Die höchste Genauigkeit erreichte MobileNetV2 mit 98%. [44]

Im Rahmen der Arbeit *Real-Time Human Pose Estimation: A MediaPipe and Python Approach for 3D Detection and Classification* wird sich mit der Klassifizierung der statischen Posen T-Haltung, Baumhaltung und Kriegerhaltung auseinandergesetzt. Zur Detektion der Schlüsselpunkte wird OpenPose verwendet, um Gelenkwinkel, Abstände und räumliche Verbindungen abzurufen. Für das Training des Posen-Erkennungsmodells wurden vortrainierte Modelle aus den OpenPose- und *MediaPipe*-Frameworks unter Verwendung von Transfer-Learning-Techniken genutzt. Die Klassifizierungsgenauigkeit beträgt bis zu 94%. [45]

4.2.3 Recurrent Neural Network

In *Proposing Posture Recognition System Combining MobilenetV2 and LSTM for Medical Surveillance* wird die Klassifizierung von sieben menschlichen Haltungen bzw. Aktivitäten untersucht, darunter Liegen, Sitzen, Hocken, Stehen, Gehen, Schlagen und Fallen. Als Eingabe für das PoseNet-Modell dient ein Video, aus dem die Schlüsselpunkte detektiert werden. Anschließend werden diese Schlüsselpunkte

[6] Transferlernen bezeichnet die Technik, ein vortrainiertes Netzwerk mithilfe zusätzlicher Daten zu verfeinern, um seine Leistungsfähigkeit auf neue Problemstellungen oder Datensätze zu steigern. [44]

an ein Long Short-Term Memory (LSTM)[7]-Modell übergeben, um die Haltungen zu klassifizieren. Das System erreicht eine Genauigkeit von bis zu 99% bei der Erkennung der verschiedenen Aktivitäten. [46]

A Study of Human Fitness Pose Classification Using Artificial Neural Networks betrachtet die Klassifizierung von zehn verschiedenen Fitnessübungen auch mit einem LSTM- und einem Gated Recurrent Unit (GRU)[8]-Modell. Dabei erreicht das LSTM-Modell eine Genauigkeit von 95% und das GRU-Modell eine Genauigkeit von 97,27%. [40]

4.3 Auswahl des Vorgehens

4.3.1 Schlüsselpunkterkennung

Zur Detektierung der Schlüsselpunkte der verschiedenen Gelenke des Schiedsrichters im Rahmen der Visual Referee Challenge haben sich MoveNet [47][48] und BlazePose [49] schon als bewährt erwiesen. BlazePose ist ein CNN, das für die Verarbeitung von RGB-Bildern als Eingabe konzipiert ist und die Koordinaten sowie Zuversichtswerte für 33 Schlüsselpunkte einer einzelnen Person in einem Bild oder Video vorhersagt. Es gibt nicht nur die x- und y- Koordinate der Schlüsselpunkte zurück, sondern auch eine virtuelle z-Koordinate. Dennoch benötigt BlazePose das Gesicht des Menschen, dessen Schlüsselpunkte erkannt werden sollen, um daraus die Position des Rumpfes ableiten zu können. [50]
MoveNet ist ebenfalls ein CNN, das RGB-Bilder als Eingabe erwartet, jedoch nur 17 Schlüsselpunkte mit x- und y-Koordinaten und Zuversichtswerten zurückgibt. Es wurde entwickelt, um robuste Ergebnisse bei der Erkennung von schnellen oder Fitness-Bewegungen mit schwierigen Posen und/oder Bewegungsunschärfe zu erzielen. Es gibt zwei Varianten des Modells: *Lightning* und *Thunder*, wobei Lightning schneller als Thunder und Thunder genauer als Lightning ist. Das Modell erfasst die Pose der Person, die dem Bildzentrum am nächsten ist, während andere Personen am Bildrahmen ignoriert werden. [51]
Obwohl BlazePose eine z-Koordinate für die Schlüsselpunkte liefert und somit potenziell zusätzliche Informationen bereitstellt, wird für das weitere Vorgehen MoveNet Thunder verwendet. Dies gewährleistet Unabhängigkeit davon, ob ein Gesicht im Frame zu sehen ist.

4.3.2 Posenerkennung

Angesichts des Klassifikationsproblems, das in dieser Arbeit behandelt wird und der Möglichkeit zur Erstellung von gelabelten Datensätzen, wird das Konzept des Supervised Learning angewendet.
Die beiden in Abschnitt 4.1 beschriebenen Ansätze zur Klassifizierung von Posen mittels Machine Learning haben den Einsatz von Deep Learning Verfahren als Zukunftsperspektive festgelegt [33][35]. Daher ist ein Deep Learning Ansatz zu bevorzugen. Insbesondere sollte dabei auf ein CNN oder RNN zurückgegriffen werden, da bei diesen Netzen im Vergleich zu herkömmlichen ANNs weniger Gewichte trainiert werden müssen [13][14]. Die Bewegungsabläufe, die bereits mit RNNs klassifiziert wurden (wie Gehen, Fallen und Schlagen [46]), sind vielfältiger und komplexer als die Schiedsrichterposen der SPL, da diese Bewegungsabläufe eine größere Bandbreite an Variationen umfassen. Die Schiedsrichterposen ähneln in ihrer Komplexität eher Yoga-Posen und sind ähnlich streng definiert [9] wie diese. Die potentielle Genauigkeit für die Klassifizierung statischer Posen mittels eines CNNs liegt bei 99,62% [43]. Es ist jedoch auch interessant zu untersuchen, wie gut CNNs mit dynamischen Posen zurechtkommen. Zudem wurden alle bisherigen Implementierungen von neuronalen Netzen auf den Naos der HTWK Leipzig unter Verwendung von CNNs durchgeführt. Daher eignet es sich, auf dieser Grundlage aufzubauen.

[7] LSTM ist eine spezielle Form von RNNs, die entworfen wurde, um langfristige Abhängigkeiten in sequenziellen Daten zu erfassen und zu verarbeiten. [38]

[8] GRU ist eine RNN-Variante, die im Vergleich zu LSTM eine einfachere Struktur hat, um langfristige Abhängigkeiten in sequentiellen Daten zu erfassen und zu verarbeiten. [38]

5 Eigene Implementierung

5.1 Datensatz

Der finale Datensatz (Tabelle 5.1) setzt sich aus drei kleineren Datensätzen zusammen. Datensatz 1 wurde vom Dutch Nao Team aufgezeichnet, wobei die Nao-Kameras verwendet wurden. Dieser Datensatz umfasst 15 Videos pro Posenklasse. Ebenfalls vom Dutch Nao Team aufgezeichnet ist Datensatz 2, jedoch mit Handykameras. Hierbei sind 30 Videos pro Posenklasse enthalten. Der dritte Datensatz wurde von mir mithilfe der Naos der HTWK aufgenommen und besteht aus 21 Videos pro Posenklasse. Alle Videos wurden im Nachhinein gespiegelt, was bedeutet, dass es pro Datensatz doppelt so viele Videos pro Posenklasse gibt.
Jedes einzelne Video hat eine Länge von 12 Sekunden und beinhaltet 29 Frames. Es wurde darauf geachtet, dass die Posen aus möglichst unterschiedlichen Winkeln aufgenommen wurden.

Datensatz	Videos pro Posenklasse
(1) Dutch Team Nao	30
(2) Dutch Team Phone	60
(3) HTWK Team Nao	42
Datensatz (1), (2) & (3)	132

Tabelle 5.1: Zusammensetzung des finalen Datensatzes

5.2 Vorverarbeitung der Daten

5.2.1 Vorverarbeitung der Videos

Um die Bilder, die mit verschiedenen Kameras aufgenommen wurden, für MoveNet Thunder vorzubereiten, müssen einige Vorverarbeitungsschritte durchgeführt werden, da MoveNet als Eingabe ein Bild mit den Dimensionen 256x256x3 erwartet. Die Dimensionen beziehen sich hierbei auf die Bildbreite in Pixeln, die Bildhöhe in Pixeln und die Werte der RGB-Farbkanäle. Als Orientierungspunkt für diese Vorverarbeitung dienen die roten Handschuhe, die der Schiedsrichter gemäß den Regeln[9] tragen muss. Zunächst werden die zwei größten Rottöne/Blobs (dargestellt in Tabelle 5.2) im Bild gesucht und aus diesen wird der Mittelpunkt berechnet. Dieser Mittelpunkt dient als Referenzpunkt für einen quadratischen Bildausschnitt. Dieser Ausschnitt wird dann auf 256x256 Pixel heruntersklaliert und als Eingabe für MoveNet bereitgestellt. Falls weniger als zwei rote Blobs gefunden werden, wird ein Standardbildausschnitt verwendet.

	lower	upper
red		
magenta		

Tabelle 5.2: Farbwerte der erkannten Rottöne

5.2.2 Erstellung der Trainings- und Validierungsdaten

MoveNet erkennt wie in Abbildung 5.1 dargestellt 17 verschiedene Schlüsselpunkte. Aufgrund der wenigen Daten werden außerdem synthetische Daten aus den Detections erstellt. Hierfür werden die ursprünglichen erfassten Schlüsselpunkte um einen zufälligen Wert zwischen 0 und 2,56 Pixeln in horizontaler und vertikaler Richtung verschoben. Dies sorgt nicht nur dafür, dass mehr Trainingsdaten zu Verfügung stehen, sondern auch, dass das Model robuster gegenüber Variationen wird.
Aus den originalen und den synthetischen Detections werden dann die Winkel zwischen den verschiedenen Gelenken berechnet. Hierzu zählen die in Tabelle 5.3 aufgelisteten Gelenke.

[9] Bei den Wettbewerben 2024 trägt der Schiedsrichter noch rote Handschuhe, allerdings könnte diese Vorgabe 2025 wegfallen [52]. Deswegen ist es wichtig, dass die roten Handschuhe für die Erkennung der Posen an sich nicht relevant ist.

Winkel an Gelenk	beteiligte Schlüsselpunkte (Abbildung 5.1)
linke Schulter	(12, 6, 8)
rechte Schulter	(7, 5, 11)
linker Ellenbogen	(10, 8, 6)
rechter Ellenbogen	(5, 7, 9)
linkes Knie	(16, 14, 12)
rechtes Knie	(11, 13, 15)
linke Hüfte	(14, 12, 6)
rechte Hüfte	(5, 11, 13)
linker Torso Schulter	(5, 6, 12)
rechter Torso Schulter	(11, 5, 6)
linker Torso Hüfte	(6, 12, 11)
rechter Torso Hüfte	(12, 11, 5)

Tabelle 5.3: Gelenke und die für die Winkelberechnung benötigten Schlüsselpunkte

Außerdem werden die Abstände der Hände zueinander und jeder Hand zur zugehörigen Schulter erfasst. Für jeden Frame werden diese extrahierten Daten in einer Comma-separated values (CSV)-Datei gespeichert, welche als Input für das neuronale Netz zur Posenerkennung dient.

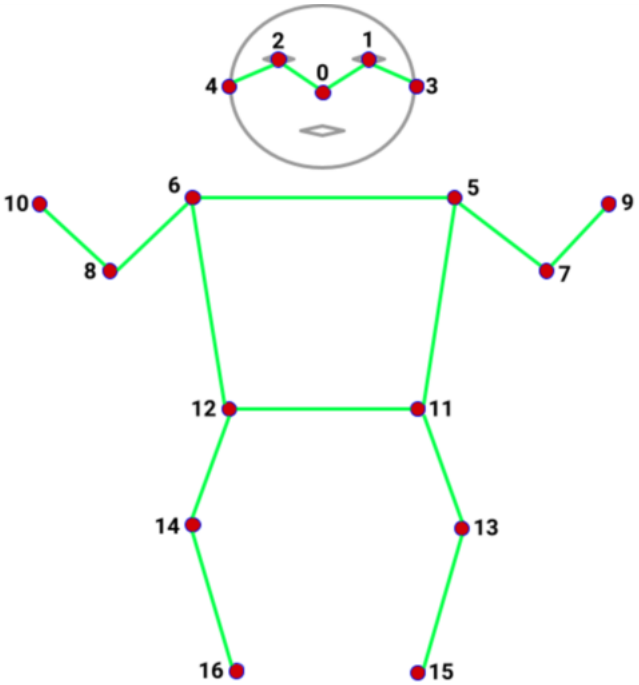


Abbildung 5.1: Erkannte Schlüsselpunkte von MoveNet: 0: Nase, 1: rechtes Auge, 2: linkes Auge, 3: rechtes Ohr, 4: linkes Ohr, 5: rechte Schulter, 6: linke Schulter, 7: rechter Ellenbogen, 8: linker Ellenbogen, 9: rechtes Handgelenk, 10: linkes Handgelenk, 11: rechte Hüfte, 12: linke Hüfte, 13: rechtes Knie, 14: linkes Knie, 15: rechter Knöchel, 16: linker Knöchel, in Anlehnung an [53]

5.3 Ansatz 1: Ein Model pro Pose

5.3.1 Datensatz

Für diesen Ansatz wird für jede Pose ein separates Modell erstellt, wobei die Klassifizierung binär erfolgt: Wird die Pose erkannt oder nicht? Dieser Ansatz führt jedoch dazu, dass für das Testen und Validieren nur ein unbalancierter Datensatz vorhanden ist, da auf jedes Beispiel für *True* zwölf Beispiele für *False* kommen. Dieses Ungleichgewicht im Datensatz kann nicht durch das Sammeln von mehr Daten behoben werden, da ein Beispiel, das für *True* in einem Modell steht, für ein anderes Modell als *False* betrachtet wird. Eine optimale Lösung besteht darin, den Datensatz im Verhältnis 50:50 aufzuteilen.
Um sich diesem Gleichgewicht annähern zu können, ohne auf zu viele Daten zu verzichten, wird eine Kombination aus Oversampling der unterrepräsentierten *True*-Klasse und Undersampling der überrepräsentierten *False*-Klasse verwendet.

Beim Oversampling werden auf jeden echten Datensatz drei synthetische Datensätze erzeugt, wie in Abschnitt 5.2.2 beschrieben. Es ist wichtig anzumerken, dass für den Validierungsdatensatz nur reale Daten verwendet werden dürfen, damit beim Training nicht die Validierungsdaten gelernt werden.
Beim Undersampling werden aus jeder Posenklasse, die repräsentativ für *False* ist, eine geringe, gleiche Anzahl zufälliger Daten ausgewählt, die dem Modelltraining zur Verfügung stehen. Dies geschieht, um den Trainings- und Validierungsdatensatz auszugleichen. Durch diesen Prozess werden die Anzahl der Daten für *True* und *False* der Trainings- und Validierungsdatensätze einander angenähert.
Die Aufteilung der Datensätze erfolgt mit einer Rate von 80% für das Training und 20% für die Validierung. Die genaue Aufteilung der Daten pro Datensatz (Datensatz 1, 2 und 3) sowie insgesamt kann den Tabellen 5.4, 5.5, 5.6 und 5.7 entnommen werden.

Datensatz 1		
	True	False
Train	15 real, 45 synth	84 real (7 per class)
Val	15 real	24 real (2 per class)

Tabelle 5.4: Zusammensetzung von Datensatz 1 (Dutch Team Nao)

Datensatz 2		
	True	False
Train	30 real, 90 synth	156 real (13 per class)
Val	30 real	36 real (3 per class)

Tabelle 5.5: Zusammensetzung von Datensatz 2 (Dutch Team Phone)

Datensatz 3		
	True	False
Train	22 real, 66 synth	120 real (10 per class)
Val	20 real	24 real (2 per class)

Tabelle 5.6: Zusammensetzung von Datensatz 3 (HTWK Team Nao)

Datensatz 1, 2 & 3		
	True	False
Train	268	360
Val	65	84

Tabelle 5.7: Zusammensetzung des Datensatzes für Ansatz 1

5.3.2 Architektur

Während der Tests mit den verschiedenen Posen hat sich die Architektur in Abbildung 5.2 als besonders effektiv erwiesen.

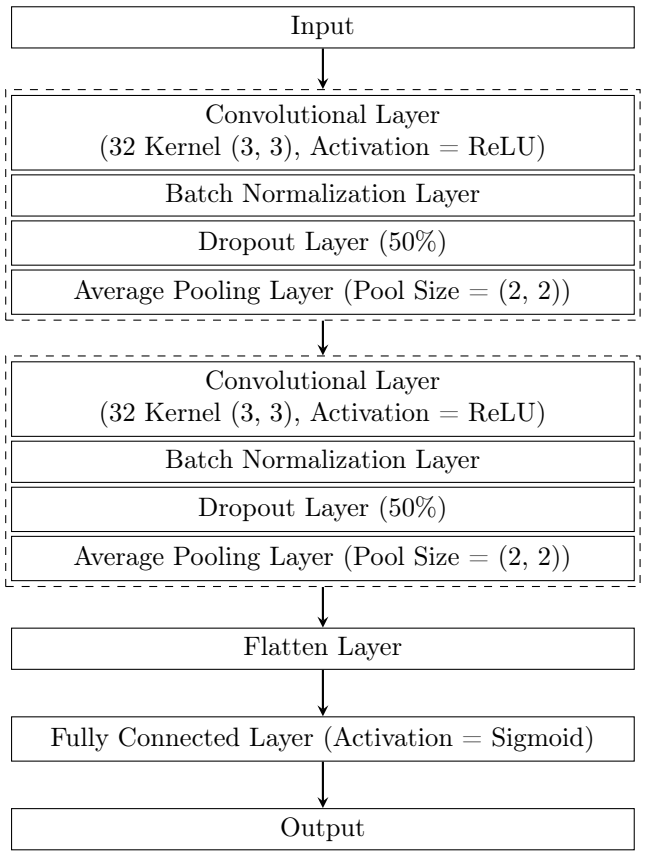


Abbildung 5.2: Modellarchitektur für jedes Modell, das für die Klassifizierung einer einzelnen Pose zuständig ist

5.3.3 Bewertung

Das Training der Modelle ist initial auf 200 Epochen[10] angesetzt. Es wird während des Trainings die Validierungsgenauigkeit beobachtet und sollte sich diese innerhalb von 30 Epochen nicht verbessern, wird der Trainingsprozess abgebrochen und das Modell mit der höchsten Validierungsgenauigkeit gespeichert.

Modell	Accuracy	Precision	Recall	F1-Score
kick in blue	96.0%	95.8%	96.3%	95.9%
kick in red	98.7%	98.5%	98.8%	98.6%
goal kick blue	97.3%	97.7%	96.9%	97.3%
goal kick red	95.3%	95.2%	95.3%	95.2%
corner kick blue	98.0%	98.0%	97.8%	97.9%
corner kick red	98.0%	97.9%	98.0%	97.9%
goal blue	91.3%	91.4%	92.1%	91.3%
goal red	92.0%	91.7%	92.2%	91.9%
full time	99.3%	99.4%	99.2%	99.3%
pushing free kick blue	94.6%	94.4%	94.7%	94.6%
pushing free kick red	94.0%	93.9%	93.8%	93.9%
player exchange blue	98.0%	98.0%	97.8%	98.0%
player exchange red	97.3%	97.2%	97.5%	97.3%
∅	96.1%	96.1%	96.2%	96.1%

Tabelle 5.8: Validierungsgenauigkeit eines jeden Modells

[10]Eine Epoche bedeutet eine Trainingsiteration über den gesamten Trainingsdatensatz.

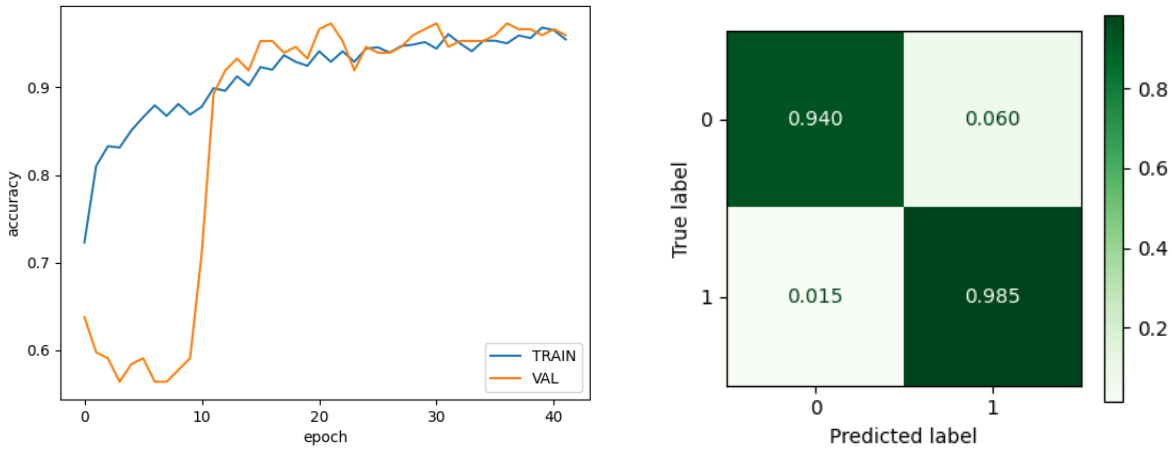


Abbildung 5.3: Trainingsergebnisse und Konfusionsmatrix für die Pose *kick in blue*

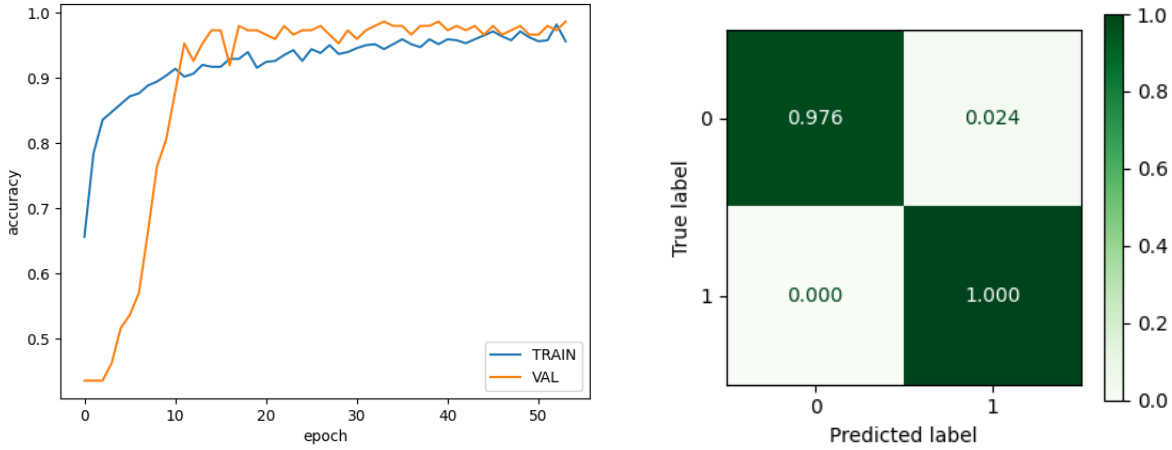


Abbildung 5.4: Trainingsergebnisse und Konfusionsmatrix für die Pose *kick in red*

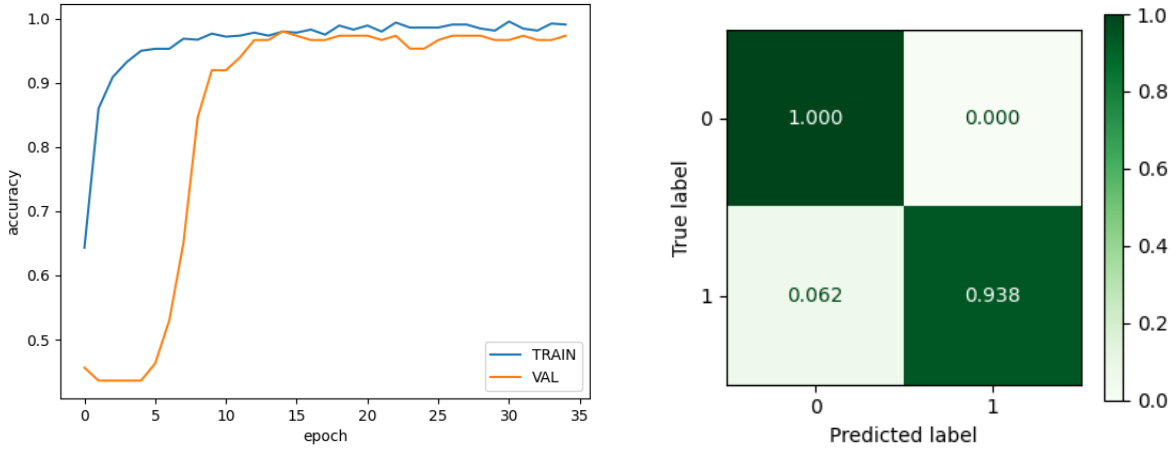


Abbildung 5.5: Trainingsergebnisse und Konfusionsmatrix für die Pose *goal kick blue*

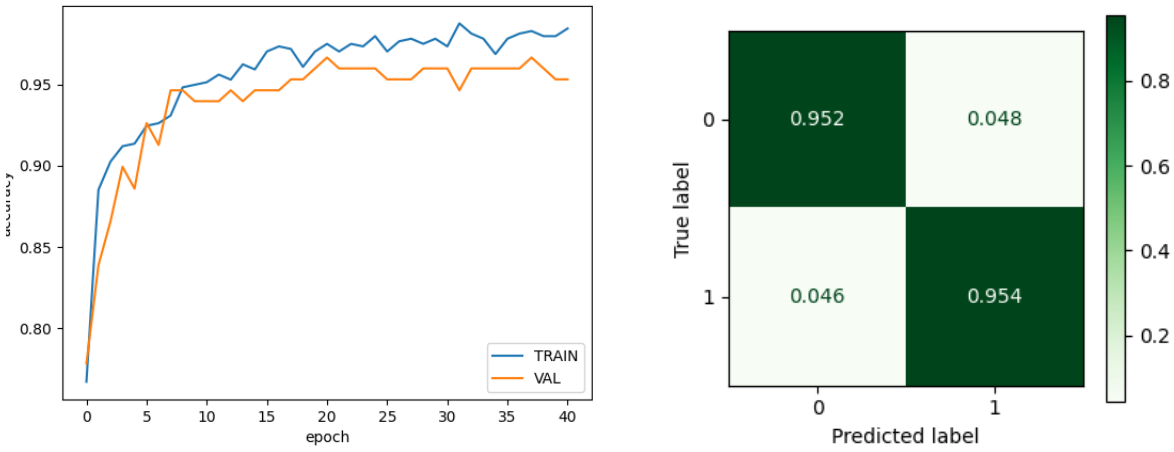


Abbildung 5.6: Trainingsergebnisse und Konfusionsmatrix für die Pose *goal kick red*

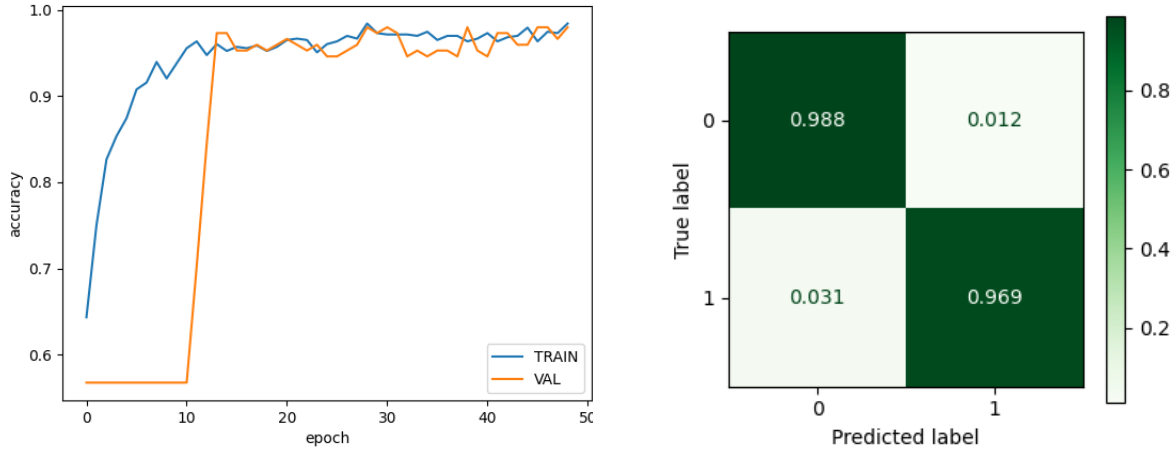


Abbildung 5.7: Trainingsergebnisse und Konfusionsmatrix für die Pose *corner kick blue*

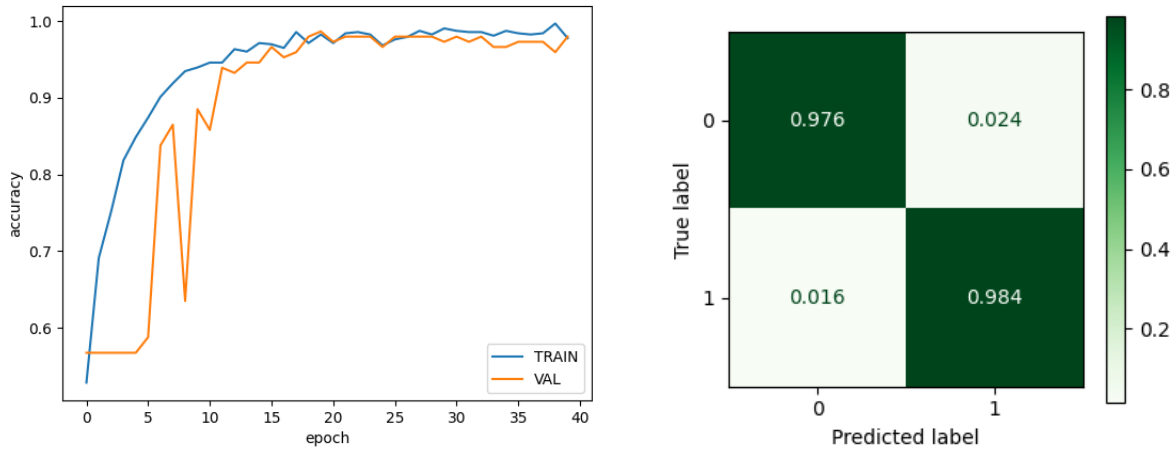


Abbildung 5.8: Trainingsergebnisse und Konfusionsmatrix für die Pose *corner kick red*

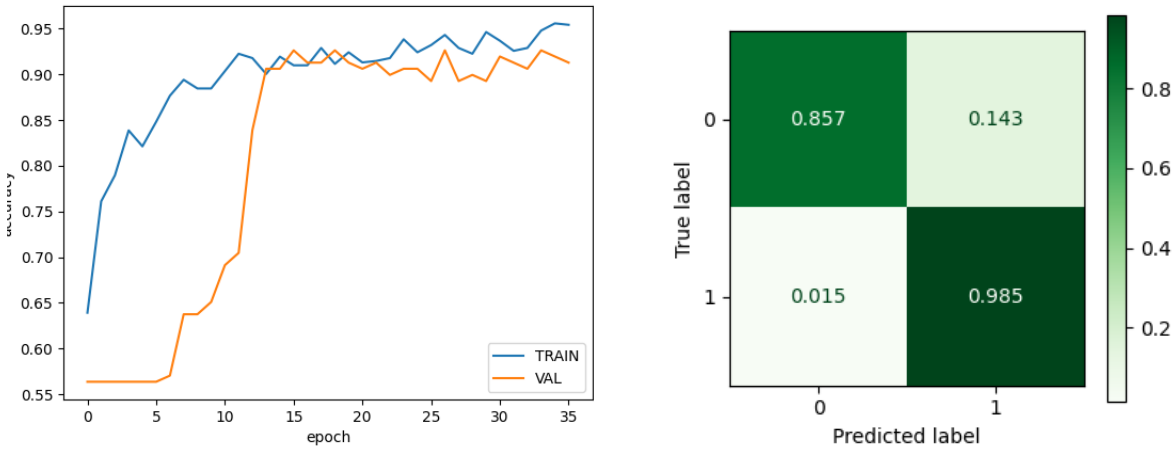


Abbildung 5.9: Trainingsergebnisse und Konfusionsmatrix für die Pose *goal blue*

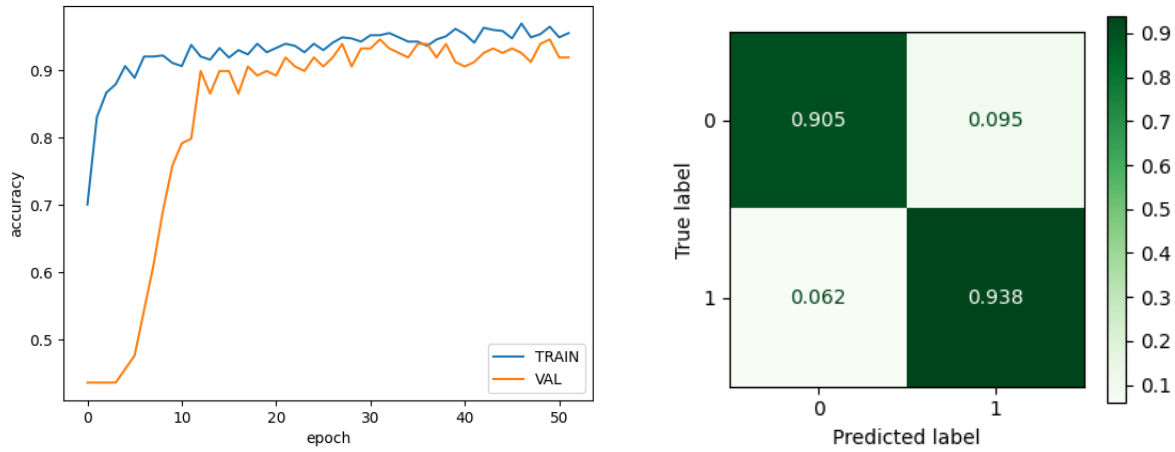


Abbildung 5.10: Trainingsergebnisse und Konfusionsmatrix für die Pose *goal red*

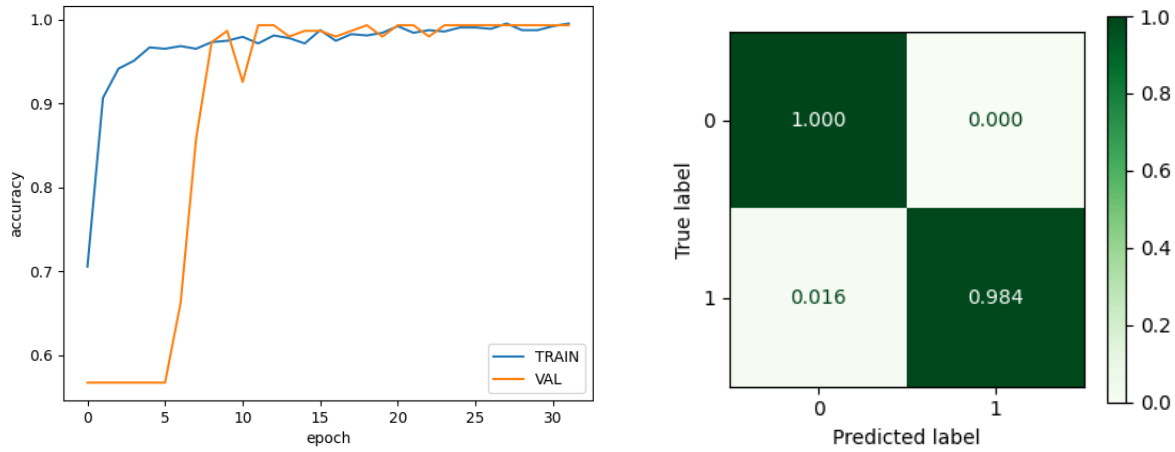


Abbildung 5.11: Trainingsergebnisse und Konfusionsmatrix für die Pose *full time*

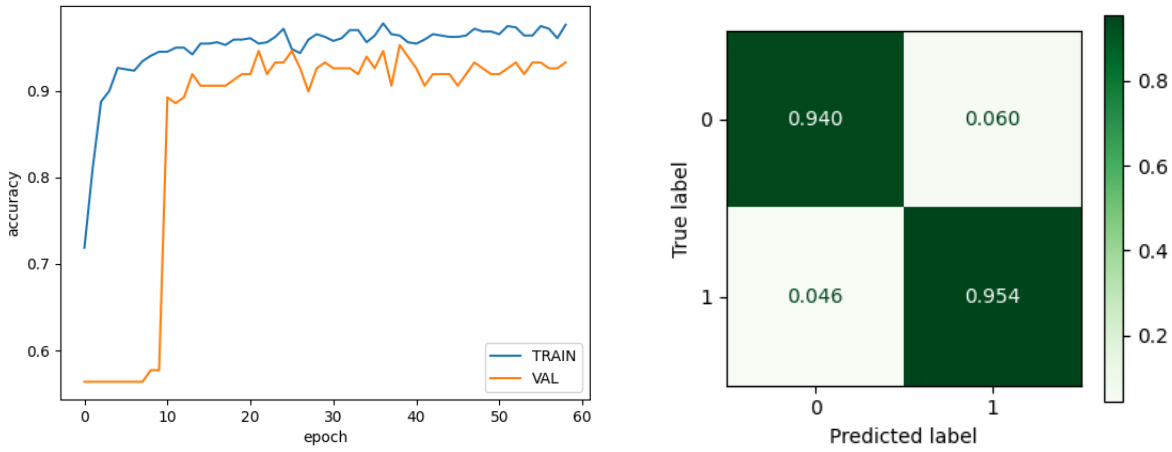


Abbildung 5.12: Trainingsergebnisse und Konfusionsmatrix für die Pose *pushing free kick blue*

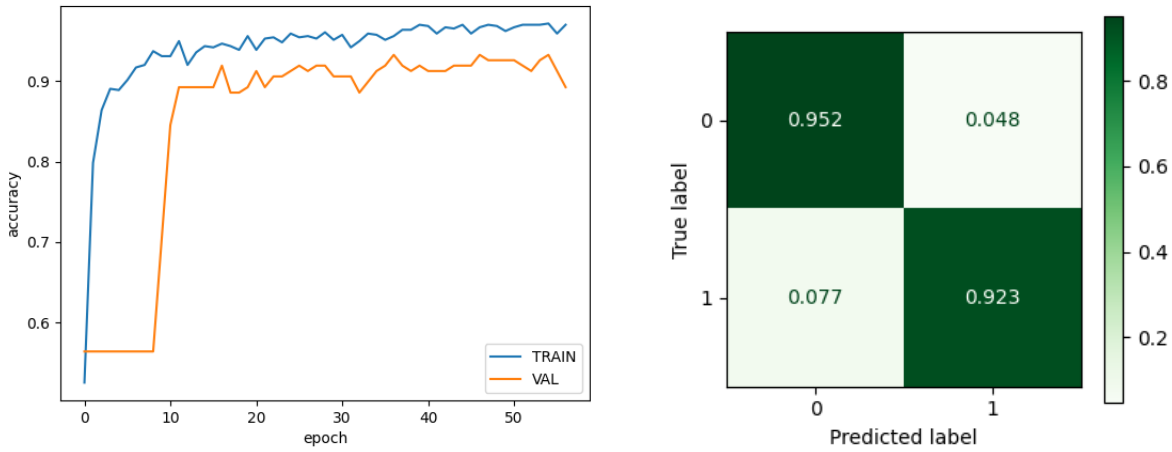


Abbildung 5.13: Trainingsergebnisse und Konfusionsmatrix für die Pose *pushing free kick red*

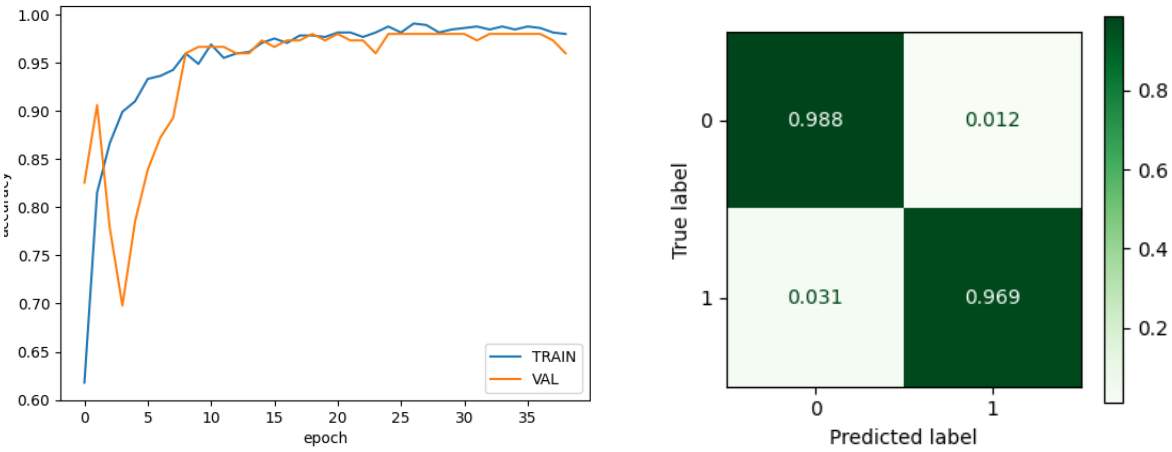


Abbildung 5.14: Trainingsergebnisse und Konfusionsmatrix für die Pose *player exchange blue*

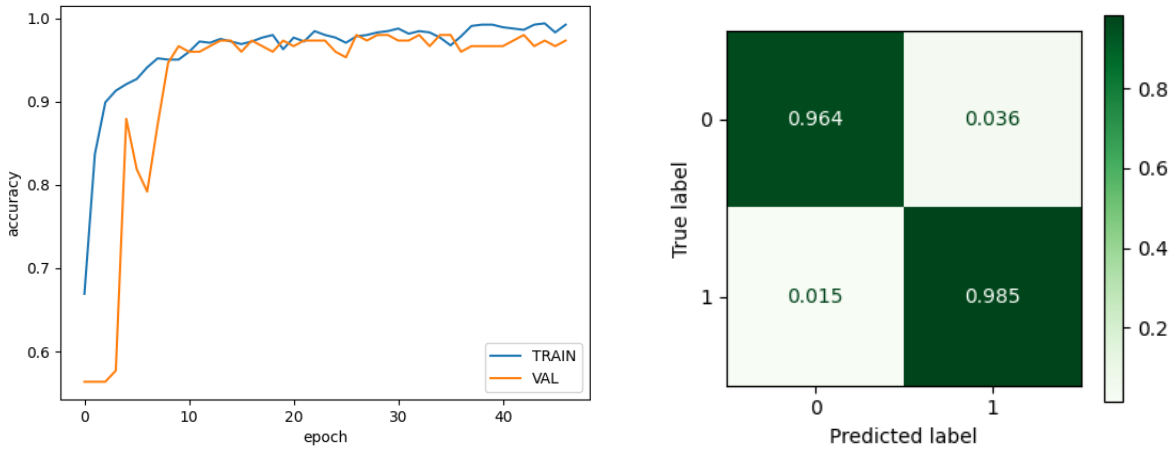


Abbildung 5.15: Trainingsergebnisse und Konfusionsmatrix für die Pose *player exchange red*

5.4 Ansatz 2: Ein Model für alle Posen

5.4.1 Datensatz

Für diesen Ansatz wird ein Modell trainiert, das alle 13 Posen gleichzeitig klassifizieren kann. Hierfür ist der bestehende Datensatz ausbalanciert, da keine Negativ-Beispiele gebraucht werden. Dennoch wird der Trainingsdatensatz mittels Oversampling erweitert, um mehr Daten zum Training zur Verfügung stehen zu haben. Die Trainings- und Validierungsdaten, die in Ansatz 2 verwendet werden, sind identisch mit denen, die in Ansatz 1 verwendet werden, um einen direkten Vergleich der beiden Ansätze gewährleisten zu können. Die Aufteilung der Datensätze erfolgt ebenfalls mit einer Rate von 80% für das Training und 20% für die Validierung. Die genaue Aufteilung der Daten kann der Tabelle 5.9 entnommen werden.

Pose	Train	Test
kick in blue/red	76 real, 228 synth	65 real
goal kick blue/red	69 real, 207 synth	65 real
corner kick blue/red	67 real, 201 synth	64 real
goal blue/red	68 real, 204 synth	65 real
pushing free kick blue/red	70 real, 210 synth	65 real
player exchange blue/red	71 real, 213 synth	65 real
full time	68 real, 204 synth	64 real

Tabelle 5.9: Zusammensetzung des Datensatzes für Ansatz 2

5.4.2 Architektur

Um die beste Architektur für das Modell, das alle Posen gleichzeitig erkennen soll, wurde *Keras Tuner* [54] verwendet. Keras Tuner ist eine Bibliothek, die von *TensorFlow* bereitgestellt wird und speziell für die Hyperparameter-Optimierung von Keras-Modellen entwickelt wurde. Hyperparameter sind die Parameter, die die Struktur und das Verhalten eines neuronalen Netzwerks beeinflussen, aber nicht durch das Training selbst gelernt werden, im Folgenden die Anzahl der Kernel in jedem Convolutional Layer, die Dropout-Rate eines jeden Dropout Layers und ob es einen dritten Layer-Stack geben wird. Ein Stack besteht aus einem Convolutional Layer mit Kernelgrößen 3x3 und ReLU-Aktivierungsfunktion, einem Batch Normalization Layer, einem Dropout Layer und einem Average Pooling Layer mit der Kernelgröße 2x2. Die Architektur (Abbildung 5.16) orientiert sich hierbei an der Modell-Architektur aus Ansatz 1 (Abbildung 5.2).
Vor der Suche nach dem besten Modell wird eingestellt, welche Suche, wie oft die Suche durchgeführt werden soll und nach welchem Kriterium das Modell bewertet wird. Hier wird die Random Search 500 Mal durchgeführt, was bedeutet, dass 500 Mal zufällige Hyperparameter in das Modell eingesetzt werden. Das beste Modell soll das mit der höchsten Validierungsgenauigkeit sein.
Die möglichen Hyperparameter sind in Tabelle 5.10 aufgelistet.

Stack	Hyperparameter
1	Kernelanzahl $k_1 \in [8, 16, 32, 64]$ Dropoutrate $d_1 \in [10, 20, 30, 40, 50]$
2	Kernelanzahl $k_2 \in [8, 16, 32, 64]$ Dropoutrate $d_2 \in [10, 20, 30, 40, 50]$
3	$add \in$ [True, False] Kernelanzahl $k_3 \in [8, 16, 32, 64]$ Dropoutrate $d_3 \in [10, 20, 30, 40, 50]$

Tabelle 5.10: Alle möglichen Hyperparameter, die Keras Tuner zur Zerfügung stehen

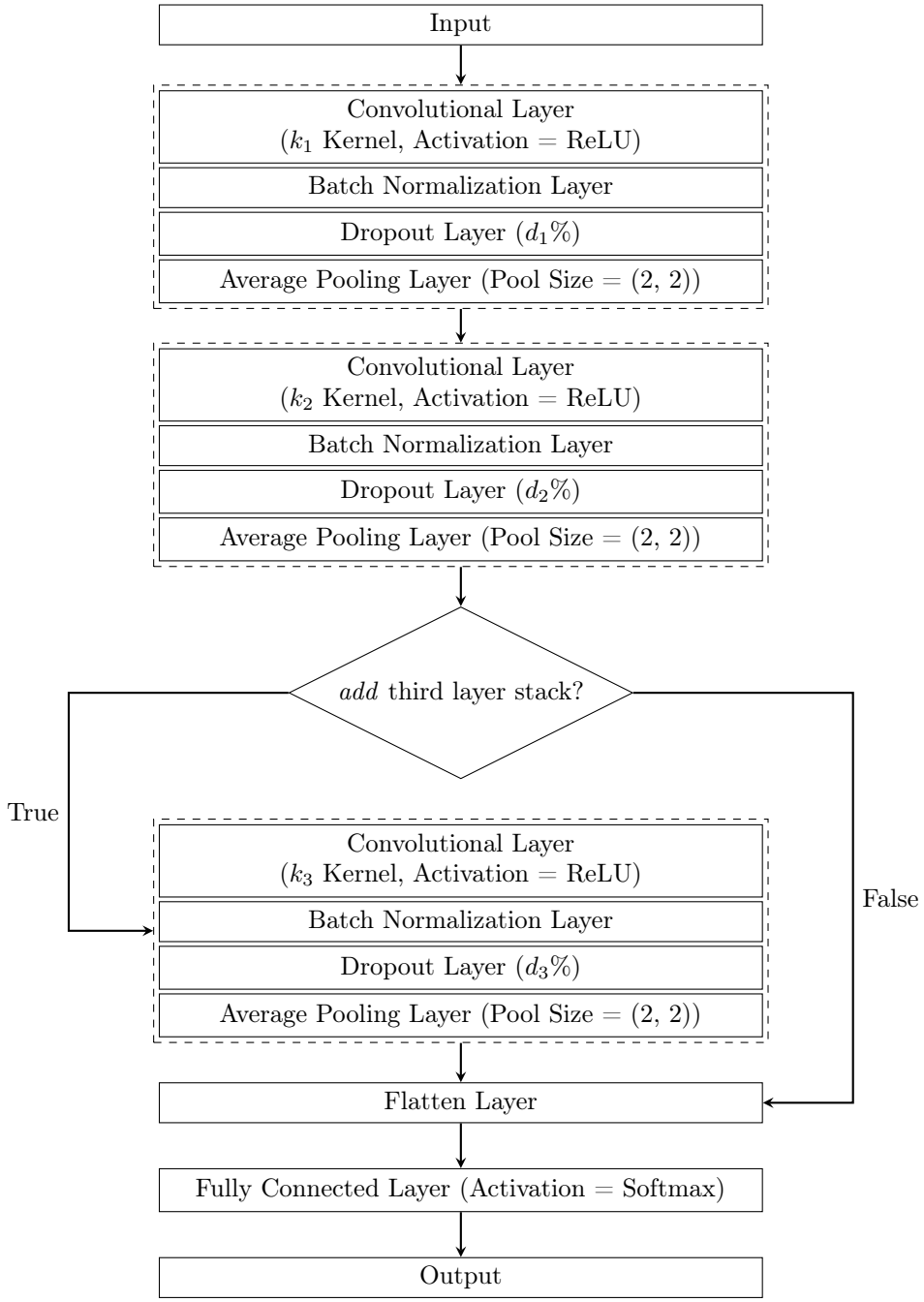


Abbildung 5.16: Modellarchitektur für die Suche mit Keras Tuner

Als die beste Modellarchitektur mit einer Accuracy von 90.90% hat sich das Modell mit den Hyperparametern aus Tabelle 5.11 herausgestellt.

Variable	Hyperparameter
k_1	64
d_1	30%
k_2	8
d_2	30%
add	True
k_3	32
d_3	30%

Tabelle 5.11: Die besten Hyperparameter, die durch die Suche mit Keras Tuner ermittelt wurden

Nach weiteren manuellen Tests hat sich jedoch die Modellarchitektur zu sehen in Abbildung 5.17 als stabiler[11] erwiesen.

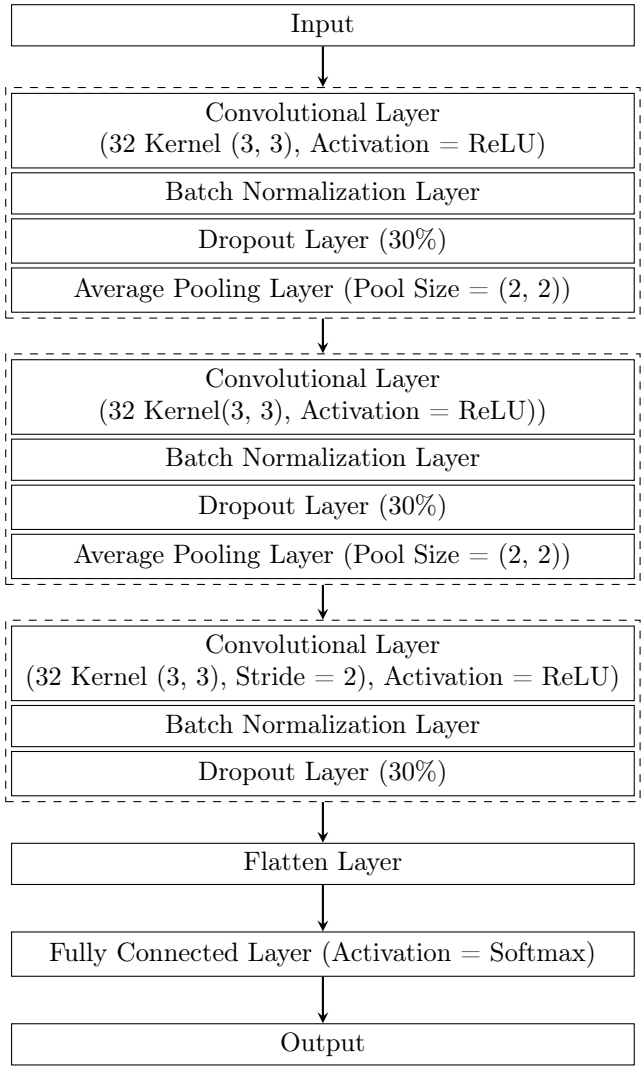


Abbildung 5.17: Modellarchitektur Version 8.2

5.4.3 Bewertung

Das Training wird hier genauso gehandhabt wie in Abschnitt 5.3.3 beschrieben. Wie in Abbildung 5.18 zu sehen, lernt das Modell bis zur siebten Epoche sehr schnell und erreicht die höchste Validierungsgenauigkeit in Epoche 38. Danach passt sich das Modell nur noch weiter an die Trainingsdaten an. Das Modell erreicht eine Accuracy von 90.9%, eine Precision von 91.3%, einen Recall von 90.9% und einen F1-Score von 90.8%.

[11] Stabiler bedeutet in diesem Kontext, dass nach mehrmaligem Training eines Modells mit dieser Architektur die Validierungsgenauigkeit ähnlich hoch war.

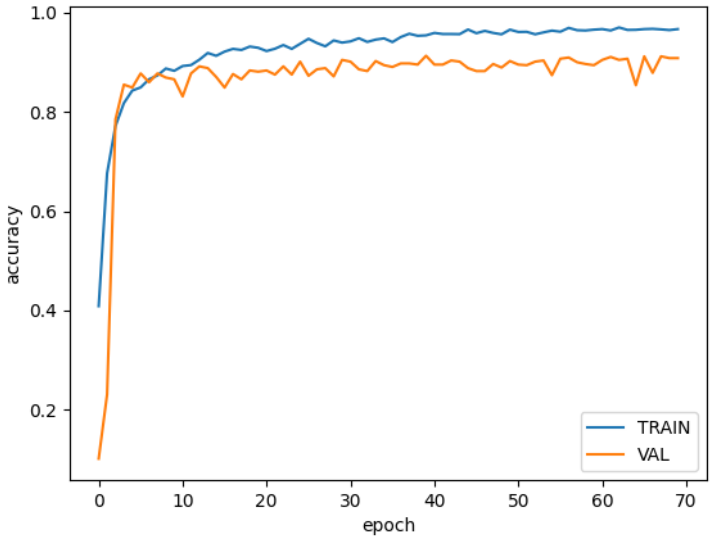


Abbildung 5.18: Trainingsergebnisse des Modells für die Klassifikation für alle 13 Posen gleichzeitig

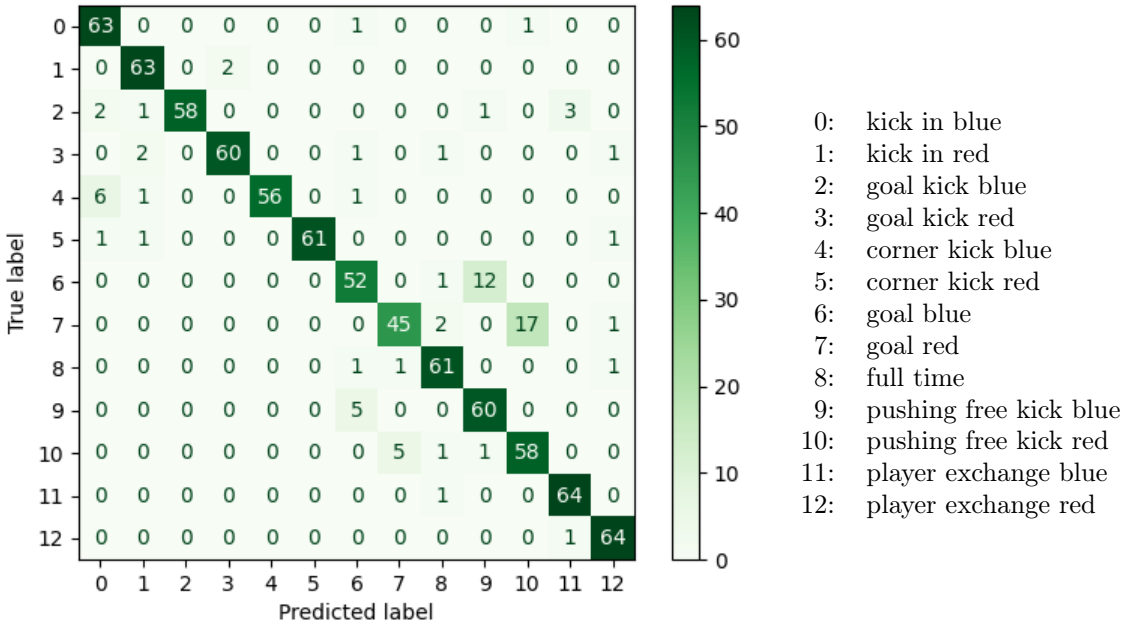


Abbildung 5.19: Konfusionsmatrix des Modells für die Klassifikation für alle 13 Posen gleichzeitig

Wie in Abbildung 5.19 zu sehen stellen sich die Posen *goal blue* und *goal red* als besonders schwierig zu erkennen heraus. Auch bei den Posen *goal kick blue/red*, *corner kick blue* und *pushing free kick blue/red* hatte das Modell Schwierigkeiten, die richtigen Klassen zuzuordnen. Jedoch ist signifikant, dass selbst wenn die falsche Pose klassifiziert wurde, meistens die richtige Seite erkannt wurde. Eine genauere Übersicht darüber ist in Tabelle 5.12 zu sehen.

Team	Richtiges Team	Falsches Team	Full Time
blue	31	3	2
red	30	4	3

Tabelle 5.12: Übersicht darüber, wie oft eine falsch erkannte Pose der richtigen Seite, der falschen Seite oder *full time* (neutral) zugeordnet wurde

5.5 Vergleich der Ansätze

Die Modelle werden auf einem ACER Aspire A515-51G ausgeführt, der mit einem Intel Core i5-8250U Prozessor betrieben wird, der über eine Taktfrequenz von 1,60 GHz, 4 physische Kerne und

8 logische Prozessoren verfügt. Klassifizierungszeit in Tabelle 5.13 beschreibt die Zeit, die das das Programm braucht, um die Posen durch alle angewandten Modelle zu klassifizieren. Floating Point Operations (flops) bezeichnet die Gleitkomma-Operationen, die das System während der Ausführung der Modelle tätigt. Accuracy, Precision, Recall und F1-Score werden in Abschnit 3.4 beschrieben. In Ansatz 1 werden die Modelle in 4 verschiedenen Threads ausgeführt, um alle Kerne des Systems auszulasten und die Laufzeit zu verringern.

	Ansatz 1	Ansatz 2
Accuracy	∅ **96.1%**	90.9%
Precision	∅ **96.1%**	91.1%
Recall	∅ **96.2%**	90.9%
F1-Score	∅ **96.1%**	90.8%
∅ Modellaufzeit	37.46 ms/step	**22.87 ms/step**
∅ Klassifizierungszeit	499 ms	**63.95 ms**
flops	291.297 (insg. 3.786.627)	**1.535.182**

Tabelle 5.13: Vergleich der beiden Lösungsansätze

Um eine Klassifizierung in Echtzeit zu gewährleisten steht dem Programm dafür ein Zeitfenster von 400 ms zur Verfügung, da danach schon der nächste Datensatz zur Auswertung bereitsteht. Ansatz 1 benötigt jedoch 499 ms, was zu viel ist. Ein Lösungsvorschlag ist, dass mit jedem Datensatz nicht jedes Modell zur Klassifizierung herangezogen wird, sondern alternierend die Hälfte, wie in Tabelle 5.14 eingeteilt. Mit dieser alternierenden Methode werden pro Datensatz im Durchschnitt nur noch 289,65 ms benötigt. Der Klassifizierungsloop kann in den drei dafür zu Verfügung stehenden Sekunden 7 mal laufen, das heißt die Pose wird mindestens drei mal von jedem Modell klassifiziert und danach kann der finale Tipp abgegeben werden.

Durchlauf 1	Durchlauf 2
kick in blue/red	goal kick blue/red
corner kick blue/red	goal blue/red
full time	full time
pushing free kick blue/red	player exchange blue/red

Tabelle 5.14: Einteilung der verwendeten Modelle

6 Fazit und Ausblick

6.1 Fazit

Das Ziel dieser Arbeit ist es, statische und dynamische Zeichen des Schiedsrichters in der Standard Platform League des RoboCups mittels CNNs in 13 verschiedene Posenklassen zu klassifizieren. In dieser Arbeit wird jede Pose durch die Schlüsselpunkte aus 29 Frames als dynamische Pose gehandhabt. Die Validierungsgenauigkeiten von Ansatz 1 mit 96,1% und Ansatz 2 mit 90,9% sind hervorragende Ergebnisse, insbesondere im Vergleich zu den in Abschnitt 4.2.2 vorgestellten Ansätzen, die ausschließlich statische und deswegen unkomplexere Posen verarbeitet haben. Diese Ergebnisse zeigen, dass CNNs auch für die Klassifizierung dynamischer Posen geeignet sind und somit die Forschungsfrage dieser Arbeit positiv beantwortet werden kann. Darüber hinaus zeigt Tabelle 5.13, dass auch in diesem Fall der typische Tradeoff zwischen Genauigkeit und Geschwindigkeit berücksichtigt werden muss. Die Verarbeitung eines Frames alle 400 ms bietet jedoch die Möglichkeit, etwas Geschwindigkeit zugunsten der Genauigkeit zu opfern.
Die in dieser Arbeit beschriebene Herangehensweise weist bestimmte Einschränkungen auf, die berücksichtigt werden müssen. Die Erkennung des Schiedsrichters aufgrund der roten Handschuhe ist eine Herausforderung, insbesondere bei Beleuchtung von oben und einer Perspektive, in der die Naos von unten schauen. Dies kann dazu führen, dass die Handschuhe manchmal sehr dunkel erscheinen. Es kann außerdem vorkommen, dass andere Gelenke als die des Schiedsrichters interpretiert werden, beispielsweise wenn sich Personen oder gegnerische Naos mit roten Shirts im Sichtfeld der eigenen Naos befinden oder wenn eine Person nahe dem Schiedsrichter steht und die Position der Handschuhe dafür sorgt, dass diese in den Bildmittelpunkt gerät. MoveNet zeigt zudem Schwierigkeiten, wenn ein Arm teilweise oder gänzlich verdeckt ist. Dies kann dazu führen, dass ein Arm fälschlicherweise als zwei erkannt wird oder dass die Pose vollständig unzureichend interpretiert wird. Naos, die sich direkt vor dem Schiedsrichter befinden und somit eine zentralere Perspektive bieten, ermöglichen eine präzisere Gelenk-Erkennung im Vergleich zu Naos, die seitlich auf den Schiedsrichter gerichtet sind.

6.2 Ausblick

Basierend auf den positiven Ergebnissen des Proof of Concept steht nun die bevorstehende Implementierung eines der beiden in Abschnitt 5 beschriebenen Ansätze auf dem Nao an, wobei dann noch Tests erforderlich sind, um sicherzustellen, dass der Nao die Posenerkennung tatsächlich wie beabsichtigt ausführt. Der erste richtige Einsatz findet im April 2024 auf der German Open des RoboCups statt und danach im Juli beim RoboCup 2024.
Da in den Wettbewerben 2024 die Visual Referee Challenge nicht mehr als Technical Challenge existiert, sondern die dabei erworbenen Skills langsam in den Hauptwettbewerb eingeführt werden, müssen die Modelle um eine Pose erweitert werden: Vor dem Beginn des Spiels zeigt der Schiedsrichter den Naos mit einer visuellen Geste an, dass sie vom `initial`[12]-Zustand in den `ready`[13]-Zustand wechseln sollen, indem er beide Hände über den Kopf hebt und diese dort für zwei Sekunden hält. Der Einsatz von anderen Posen wird im aktuellen vorläufigen Regelbuch nicht beschrieben. [52]
Bis zu den Wettbewerben können weitere Test erfolgen, wie die Posenerkennung optimiert werden kann. Ein Vorschlag ist, die Vorverarbeitung der Daten zu optimieren, indem potentiell unwichtige Schlüsselpunkte, wie zum Beispiel die Knie und Füße nicht mit in das weitere Vorgehen einbezogen werden. Darüber hinaus kann auch die Normalisierung der restlichen Schlüsselpunkte in Betracht gezogen werden.
Um den Laufzeitproblemen in Ansatz 1 zuvorzukommen bietet es sich an, hybride Modelle, die zwei Posen (*<pose>_red* und *<pose>_blue*) gleichzeitig klassifizieren können, zu trainieren. Dies reduziert die Anzahl der Modelle von dreizehn auf sieben. Alternativ kann man für die statischen Posen Modelle trainieren, die nur die Daten eines Frames als Eingabe benötigen. Dies könnte die Laufzeit für ein einzelnes Modell signifikant verringern und erhöht die Anzahl der Versuche in denen der Nao die Pose erkennen kann. Für die dynamischen Posen kann man überprüfen, ob auch weniger als 29 Frames für die Erkennung dieser nötig sind. Dazu muss geprüft werden, ob die Erkennungsrate der richtigen Pose in weniger als zwölf Sekunden signifikant steigt. Erste Tests mit der Pose *Full Time* und dem Modell aus Ansatz 1 haben ergeben, dass das Modell nur acht statt zwölf Sekunden benötigt, um sicher zu klassifizieren, ob die Pose vorliegt oder nicht.

[12] `initial` - In desem Zustand stehen die Roboter aufrecht und dürfen sich nicht bewegen. [52]
[13] `ready` - In diesem Zustand gehen die Roboter in ihre Position für den Anstoß oder einen Elfmeter. [52]

Literatur

[1] „Household Robots Market - Size, Growth - Industry Report“. (), Adresse: https://www.mordorintelligence.com/industry-reports/household-robots-market (besucht am 30. 03. 2024).

[2] „World Robotics 2023 Report: Asia ahead of Europe and the Americas“, IFR International Federation of Robotics. (26. Sep. 2023), Adresse: https://ifr.org/ifr-press-releases/news/world-robotics-2023-report-asia-ahead-of-europe-and-the-americas (besucht am 30. 03. 2024).

[3] M. Hasenbein, „Mensch-KI-Interaktion und Mensch-Roboter-Interaktion“, in *Mensch und KI in Organisationen: Einfluss und Umsetzung Künstlicher Intelligenz in wirtschaftspsychologischen Anwendungsfeldern*, Springer, 2023, S. 57–83, ISBN: 978-3-662-66375-2. DOI: 10.1007/978-3-662-66375-2_5.

[4] M. Guertler, L. Tomidei, N. Sick u. a., „When is a robot a cobot? Moving beyond manufacturing and arm-based cobot manipulators“, *Proceedings of the Design Society*, Jg. 3, S. 3889–3898, Juli 2023, ISSN: 2732-527X. DOI: 10.1017/pds.2023.390.

[5] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda und E. Osawa, „RoboCup: The Robot World Cup Initiative“, in *Proceedings of the first international conference on Autonomous agents - AGENTS '97*, Marina del Rey, California, United States: ACM Press, 1997, S. 340–347, ISBN: 978-0-89791-877-0. DOI: 10.1145/267658.267738.

[6] A. Ferrein und G. Steinbauer, „20 Years of RoboCup“, *KI - Künstliche Intelligenz*, Jg. 30, Nr. 3, S. 225–232, 1. Okt. 2016, ISSN: 1610-1987. DOI: 10.1007/s13218-016-0449-5.

[7] „RoboCup Objective“. (), Adresse: https://www.robocup.org/objective (besucht am 02. 01. 2024).

[8] *RoboCup Standard Platform League (NAO) Rule Book 2023*. Adresse: https://spl.robocup.org/wp-content/uploads/SPL-Rules-2023.pdf (besucht am 02. 01. 2024).

[9] *RoboCup Standard Platform League (NAO) Technical Challenges 2023*. Adresse: https://spl.robocup.org/wp-content/uploads/SPL-Challenges-2023.pdf (besucht am 02. 01. 2024).

[10] „HTWK Robots Explained“, HTWK Leipzig Robots. (), Adresse: https://mediaserver.htwk-leipzig.de/permalink/v1261b237659bqoa2b8z/ (besucht am 28. 03. 2024).

[11] O. A. Montesinos López, A. Montesinos López und J. Crossa, „Fundamentals of Artificial Neural Networks and Deep Learning“, in *Multivariate Statistical Machine Learning Methods for Genomic Prediction*, Cham: Springer International Publishing, 2022, S. 379–425, ISBN: 978-3-030-89010-0. DOI: 10.1007/978-3-030-89010-0_10.

[12] N. Gupta, N. VAYA und B. MISHRA, „Artificial Neural Network“, *Eastern pharmacist*, Jg. 40, Nr. 475, S. 39–41, 2013.

[13] K. O'Shea und R. Nash, „An Introduction to Convolutional Neural Networks“, *CoRR*, Jg. abs/1511.08458, 2015. DOI: 10.48550/arXiv.1511.08458.

[14] S. Albawi, T. A. Mohammed und S. Al-Zawi, „Understanding of a convolutional neural network“, in *2017 International Conference on Engineering and Technology (ICET)*, Aug. 2017, S. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[15] L. Mosser, O. Dubrule und M. Blunt, „Stochastic Reconstruction of an Oolitic Limestone by Generative Adversarial Networks“, *Transport in Porous Media*, Jg. 125, Okt. 2018. DOI: 10.1007/s11242-018-1039-9.

[16] L. Zhao und Z. Zhang, „A improved pooling method for convolutional neural networks“, *Scientific Reports*, Jg. 14, Nr. 1, S. 1589, 18. Jan. 2024, ISSN: 2045-2322. DOI: 10.1038/s41598-024-51258-6.

[17] R. Shyam, „Convolutional neural network and its architectures“, *Journal of Computer Technology & Applications*, Jg. 12, Nr. 2, S. 6–14, 2021. DOI: 10.37591/JoCTA.

[18] S. Sharma, S. Sharma und A. Athaiya, „Activation functions in neural networks“, *Towards Data Sci*, Jg. 6, Nr. 12, S. 310–316, 2017. DOI: 10.33564/IJEAST.2020.v04i12.054.

[19] K. Banerjee, V. P. C., R. R. Gupta, K. Vyas, A. H. und B. Mishra, „Exploring Alternatives to Softmax Function“, *CoRR*, Jg. abs/2011.11538, 2020. DOI: 10.48550/arXiv.2011.11538.

[20] S. Ioffe und C. Szegedy, „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift“, *CoRR*, Jg. abs/1502.03167, 2015. DOI: 10.48550/arXiv.1502.03167.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever und R. Salakhutdinov, „Dropout: a simple way to prevent neural networks from overfitting“, *The Journal of Machine Learning Research*, Jg. 15, Nr. 1, S. 1929–1958, 1. Jan. 2014, ISSN: 1532-4435. Adresse: https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf.

[22] A. Dongare, R. Kharde, A. D. Kachare u. a., „Introduction to artificial neural network“, *International Journal of Engineering and Innovative Technology (IJEIT)*, Jg. 2, Nr. 1, S. 189–194, 2012. Adresse: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=04d0b6952a4f0c7203577afc9476c2fcab2cba06.

[23] H. U. Dike, Y. Zhou, K. K. Deveerasetty und Q. Wu, „Unsupervised Learning Based On Artificial Neural Network: A Review“, in *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, 2018, S. 322–327. DOI: 10.1109/CBS.2018.8612259.

[24] Y. Li, „Deep Reinforcement Learning: An Overview“, *CoRR*, Jg. abs/1701.07274, Nr. arXiv:1701.07274, 26. Nov. 2018. DOI: 10.48550/arXiv.1701.07274.

[25] A. Taner, Y. B. Öztekin und H. Duran, „Performance analysis of deep learning CNN models for variety classification in hazelnut“, *Sustainability*, Jg. 13, Nr. 12, S. 6527, 2021. DOI: 10.3390/su13126527.

[26] Google, *MoveNet*, https://www.kaggle.com/models/google/movenet, 1. Apr. 2021.

[27] TensorFlow, *PoseNet*, https://www.kaggle.com/models/tensorflow/posenet-mobilenet, 6. Okt. 2020.

[28] MediaPipe, *BlazePose*, https://www.kaggle.com/models/mediapipe/blazepose, 19. Mai 2021.

[29] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei und Y. Sheikh, „OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields“, *CoRR*, Jg. abs/1812.08008, 2019. DOI: 10.48550/arXiv.1812.08008.

[30] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo und R. Girshick, *Detectron2*, https://github.com/facebookresearch/detectron2, 2019.

[31] B. Mahesh, „Machine Learning Algorithms - A Review“, *International Journal of Science and Research (IJSR).[Internet]*, Jg. 9, Nr. 1, S. 381–386, 2020, ISSN: 2319-7064. DOI: 10.21275/ART20203995.

[32] L. Breiman, „Random Forests“, *Machine Learning*, Jg. 45, Nr. 1, S. 5–32, 1. Okt. 2001, ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

[33] R. Chatterjee, S. Roy, S. H. Islam und D. Samanta, „An AI Approach to Pose-based Sports Activity Classification“, in *2021 8th International Conference on Signal Processing and Integrated Networks (SPIN)*, 2021, S. 156–161. DOI: 10.1109/SPIN52536.2021.9565996.

[34] P. K. Syriopoulos, N. G. Kalampalikis, S. B. Kotsiantis und M. N. Vrahatis, „kNN Classification: a review“, *Annals of Mathematics and Artificial Intelligence*, 1. Sep. 2023, ISSN: 1573-7470. DOI: 10.1007/s10472-023-09882-x.

[35] D. Shah, V. Rautela, C. Sharma und A. Florence A, „Yoga Pose Detection Using Posenet and k-NN“, in *2021 International Conference on Computing, Communication and Green Engineering (CCGE)*, 2021, S. 1–4. DOI: 10.1109/CCGE50943.2021.9776451.

[36] Google. „Optionen für die Klassifizierung von Positionen | ML Kit | Google for Developers“, ML Kit. (2. Dez. 2022), Adresse: https://developers.google.com/ml-kit/vision/pose-detection/classifying-poses?hl=de (besucht am 08. 03. 2024).

[37] P. P. Shinde und S. Shah, „A Review of Machine Learning and Deep Learning Applications“, in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2018, S. 1–6. DOI: 10.1109/ICCUBEA.2018.8697857.

[38] F. M. Shiri, T. Perumal, N. Mustapha und R. Mohamed, „A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU“, *arXiv preprint arXiv:2305.17473*, 2023.

[39] V. Anand Thoutam, A. Srivastava, T. Badal u. a., „Yoga Pose Estimation and Feedback Generation Using Deep Learning“, *Computational Intelligence and Neuroscience*, Jg. 2022, V. Kumar, Hrsg., S. 4 311 350, 24. März 2022, Publisher: Hindawi, ISSN: 1687-5265. DOI: 10.1155/2022/4311350.

[40] S. Shang, R. Jin und K. Desai, „A Study of Human Fitness Pose Classification Using Artificial Neural Networks“, in *2023 International Conference on Information Technology (ICIT)*, 2023, S. 250–255. DOI: 10.1109/ICIT58056.2023.10225860.

[41] „Human pose classification with MoveNet and TensorFlow lite“, TensorFlow. (15. Mai 2023), Adresse: https://www.tensorflow.org/lite/tutorials/pose_classification (besucht am 11. 03. 2024).

[42] N. F. Janbi und N. Almuaythir, „BowlingDL: A Deep Learning-Based Bowling Players Pose Estimation and Classification“, in *2023 1st International Conference on Advanced Innovations in Smart Cities (ICAISC)*, 2023, S. 1–6. DOI: 10.1109/ICAISC56366.2023.10085434.

[43] S. Garg, A. Saxena und R. Gupta, „Yoga pose classification: a CNN and MediaPipe inspired deep learning approach for real-world application“, *Journal of Ambient Intelligence and Humanized Computing*, Jg. 14, Nr. 12, S. 16 551–16 562, 2023. DOI: 10.1007/s12652-022-03910-0.

[44] K. A. Tanjaya, M. F. Naufal, H. Arwoko und T. Informatika, „Pilates pose classification using MediaPipe and convolutional neural networks with transfer learning“, *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, Jg. 9, Nr. 2, 2023, ISSN: 2338-3070.

[45] S. Negi, M. Garg, H. Maindola, V. Kansal, U. Jain und S. Bhatla, „Real-Time Human Pose Estimation: A MediaPipe and Python Approach for 3D Detection and Classification“, in *2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS)*, 2023, S. 128–133. DOI: 10.1109/ICTACS59847.2023.10390506.

[46] P. Nguyen Huu, N. Nguyen Thi und T. P. Ngoc, „Proposing Posture Recognition System Combining MobilenetV2 and LSTM for Medical Surveillance“, *IEEE Access*, Jg. 10, S. 1839–1849, 2022. DOI: 10.1109/ACCESS.2021.3138778.

[47] „In-Game Visual Referee Challenge - B-Human“, B-Human. (12. Okt. 2023), Adresse: https://docs.b-human.de/master/challenges/in-game-visual-referee-challenge/ (besucht am 12. 03. 2024).

[48] D. X. Catarrinho, F. Nagelhout, L. van Iterson und N. Scholten, „Report- Visual Referee Challenge“, Techn. Ber., 2023. Adresse: https://staff.fnwi.uva.nl/a.visser/research/nao/2023/Visual_Referee_Challenge_Project_Report.pdf (besucht am 12. 03. 2024).

[49] D. Vermaas, „RoboCup The Visual Referee Challenge“, 1. Juli 2022. Adresse: https://staff.fnwi.uva.nl/a.visser/education/bachelorAI/daniel_vermaas_thesis_bsc.pdf (besucht am 12. 03. 2024).

[50] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang und M. Grundmann, „BlazePose: On-device Real-time Body Pose tracking“, *CoRR*, Jg. abs/2006.10204, 2020. DOI: 10.48550/arXiv.2006.10204.

[51] Y.-H. Chen, R. Votel, F. Beletti und A. Oerlemans, *MoveNet ModelCard*. Adresse: https://storage.googleapis.com/movenet/MoveNet.SinglePose%20Model%20Card.pdf (besucht am 16. 01. 2024).

[52] *RoboCup Standard Platform League (NAO) Rule Book 2024*. Adresse: https://spl.robocup.org/wp-content/uploads/SPL-Rules-master.pdf (besucht am 28. 03. 2024).

[53] *MoveNet Keypoints*, TensorFlow GitHub, 9. März 2022. Adresse: https://github.com/tensorflow/tfjs-models/blob/master/pose-detection/README.md (besucht am 03. 04. 2024).

[54] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi u. a., *KerasTuner*, https://github.com/keras-team/keras-tuner, 2019.

A Anhang

A.1 RoboCup Standard Platform League (NAO) Technical Challenges

2 In-Game Visual Referee Challenge

2.1 Challenge Goal

In the current SPL rules, the only time that a robot is required to listen directly to the human referees is for the kick-off and goal whistle. Otherwise, all human referee decisions are communicated to the robots via electronic GameController messages. In moving towards the 2050 RoboCup goal, robots will need to directly interpret referee calls and signals (such as whistles, spoken calls and hand signals), rather than receive information from an external electronic source. Building on the Visual Referee Challenge from 2022, this year's In-game Visual Referee Challenge asks the robots to detect visual referee signs during a regular match when a whistle has been blown. The normal game play is not influenced by this challenge.

This technical challenge tests a robot's ability to identify three categories of hand signals during a match:

1. Static hand signals with one hand.
2. Static hand signals with two hands.
3. Dynamic (motion) hand signals with one or two hands.

The intent of this challenge is to choose a *subset* of potential referee calls in SPL matches and test ability of a team to recognize different types of hand signals in preparation for adoption in future RoboCup matches.

2.2 Challenge Setup

As this is an in-game challenge, the challenge is executed during all preliminary matches of the main competition. All usual rules apply and teams are free to participate or not.

Two additional assistants are needed for this challenge: One of them is standing on the border strip at the T-junction of the halfway line with the touchline opposite the technical area, wearing referee clothes and red gloves. The purpose of this clothing is to clearly distinguish the challenge assistant and its hands from the other referees and from people in the background. The other assistant is sitting in the technical area and has an ordered list of hand-signals (see Section 2.3) that is obtained from the organizers before the match starts. Each list contains two random permutations of the 13 hand-signals (i. e. 26 items in total, all hand-signals appear once within the first 13 items and a second time in the other 13 items). It is guaranteed that hand-signals from all three categories appear within the first four items of the list.

During `set` and anytime during `playing` when the head referee whistles (kick-off, goal, end of half) except for penalty kicks, the challenge assistant in the technical area will communicate the next hand-signal from the list to the challenge assistant on the field. Starting $5\,\mathrm{s}$ after the whistle, the challenge assistant on the field will perform that hand-signal for $10\,\mathrm{s}$. In case the head referee has to stand on the T-junction, he will stand close to the challenge assistant and do his referee job from that close by position.

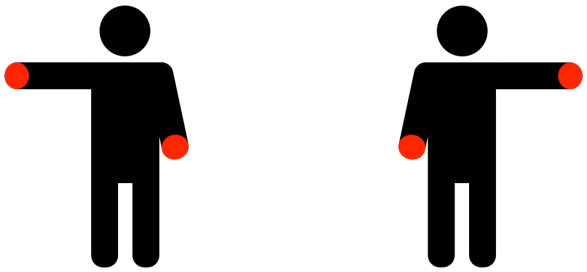
The description of this challenge and hand-signals are described based on the viewpoint of the challenge assistant. In these descriptions from the perspective of the head referee the "red team" is defined as playing from left-to-right, and the "blue team" as playing from right-to-left. The use of colors for identifying teams is used to give equivalence to the head referee calls during the match.

The challenged team reports its evaluation of the particular hand-signal to the GameController using the protocol described at `https://github.com/RoboCup-SPL/GameController3/blob/master/game_controller_msgs/headers/VisualRefereeChallenge.h`. Only reports within $15\,\mathrm{s}$ after the signaling started (i. e. $20\,\mathrm{s}$ after the whistle) will be accepted. If multiple reports arrive, only the first one will be counted.

It is possible that two events that require a hand-signal to be shown occur within a short amount of time (e. g. a goal after a kick-off or the end of half after a goal or kick-off). The assistant in the technical area should always proceed in the list of hand-signals, even if the assistant on the field is still showing the previous signal. The assistant on the field should attempt to perform the hand-signal only if they are completely done (i. e. $10\,\mathrm{s}$ have passed) with the previous signal. During evaluation however, any event that the GameController believes to be within $30\,\mathrm{s}$ of the previous event will be ignored.

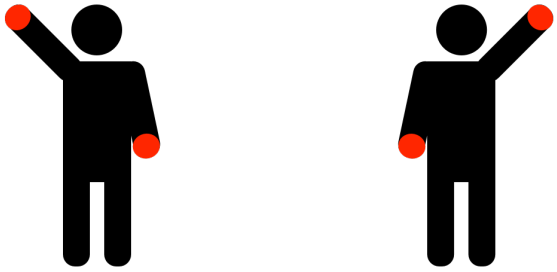
2.3 Available Hand-Signals

Each hand-signal for the challenge is **described from the perspective of the head referee** and **pictured from the perspective of the robots**. Note that for the purpose of clarity, these do not necessarily correspond to human soccer hand-signals.



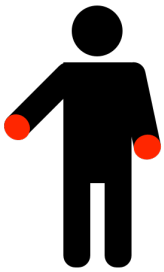
(a) Kick-in ⟨blue⟩ Team (b) Kick-in ⟨red⟩ Team

Figure 1: **Kick-in ⟨color⟩ Team.** One-handed signal. One arm, extended horizontally in the direction of the half of the field corresponding to the team that receives the Kick-in Free Kick. That is, right arm extended for the "Blue team", and left arm extended for the "Red team". The non-signal hand is flat and motionless by the side of the body.



(a) Goal Kick ⟨blue⟩ Team (b) Goal Kick ⟨red⟩ Team

Figure 2: **Goal Kick ⟨color⟩ Team.** One-handed signal. One arm, extended 45-degree *up* in the direction of the end of the field where the goal kick will occur. That is, right arm extended for the "Blue team", and left arm extended for the "Red team". The non-signal hand is flat and motionless by the side of the body.

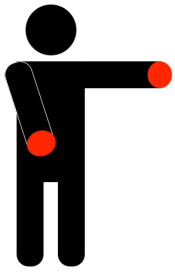


(a) Corner Kick ⟨blue⟩ Team (on the half of the red team)

(b) Corner Kick ⟨red⟩ Team (on the half of the blue team)

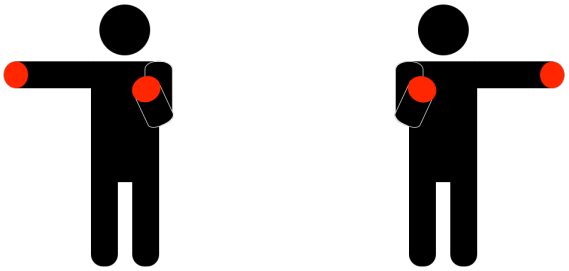
Figure 3: **Corner Kick ⟨color⟩ Team.** One-handed signal. One arm, extended 45-degree *down* in the direction of the team executing the corner kick. That is, right arm extended for the "Blue team" executing the corner kick on the "Red team's" side, and left arm extended for the "Red team" executing the corner kick on the "Blue team's" side. The non-signal hand is flat and motionless by the side of the body.

(a) Goal ⟨blue⟩ Team



(b) Goal ⟨red⟩ Team

Figure 4: **Goal ⟨color⟩ Team.** Two-handed signal. One arm, extended pointing at the center circle. Other arm, extended horizontally in the direction of the half of the field corresponding to the team that scored the goal. That is, right arm extended for the "Blue team", and left arm extended for the "Red team".



(a) Pushing Free-kick ⟨blue⟩ Team because a red robot has pushed.

(b) Pushing Free-kick ⟨red⟩ Team because a blue robot has pushed.

Figure 5: **Pushing Free-kick ⟨color⟩ Team.** Two-handed signal. One arm, vertical with bent elbow and palm facing in the direction of the extended arm. Other arm, extended horizontally in the direction of the half of the field corresponding to the team that is *executing* the Free-kick. That is, left arm extended for the "Red team", and right arm extended for the "Blue team".

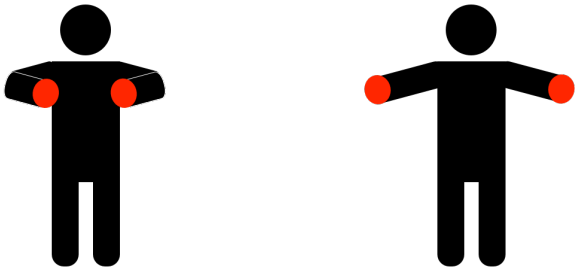
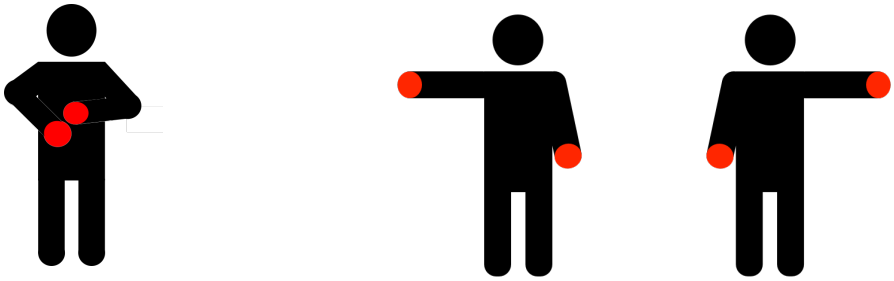


Figure 6: **Full-Time.** Dynamic two-handed signal. Both arms slowly move symmetrically inward and outwards on a horizontal plane, bending at the elbows.



(a) Rotation around virtual axis to indicate substitution.

(b) Kick-in signal to indicate the team which is substituting a player. Two possible directions.

Figure 7: **Substitution.** Dynamic two-handed signal. Both arms slowly rotate symmetrically around a virtual horizontal axis parallel to the touchline. After three rotations the referee indicates the team which is substituting a player with the kick-in signal.

2.4 Challenge Evaluation

A team scores 1 point for every direction (team blue, team red or none) and 1 point for every hand-signal (ignoring the direction) correctly identified. The time it takes a team to report a hand-signal (relative to the when signaling started, as calculated by the GameController) is recorded. If a team does not report any hand-signal within the time limit, the time for the hand-signal is $15\,\mathrm{s}$. For incorrect reports, the time is how long that incorrect report took.

As the head referee only whistles in three different match situations (kick-off, goal, end of half), the signals are categorized accordingly. The average points and time per category will be calculated. The average points $\bar{P}_{\mathrm{XXX}}$ are summed with weighting factors as well as the average time $\bar{T}_{\mathrm{XXX}}$:

$$\Sigma_{\mathrm{Points}} = \bar{\mathrm{P}}_{\mathrm{kick\_off}} + 3 \times \bar{\mathrm{P}}_{\mathrm{goal}} + 2 \times \bar{\mathrm{P}}_{\mathrm{end\_of\_half}}$$
$$\Sigma_{\mathrm{Time}} = 3 \times \bar{\mathrm{T}}_{\mathrm{kick\_off}} + \bar{\mathrm{T}}_{\mathrm{goal}} + 2 \times \bar{\mathrm{T}}_{\mathrm{end\_of\_half}}$$

These two values will be determined for each match. After the preliminary matches have been finished, the minimum number of matches each team has played will be evaluated over all teams referenced as N. For each team the total number of points and the total time is calculated as the sum of the $N - 1$ best ranked matches according to the number of points.

Teams are ranked by their total number of points. In the case of a tie, the team with the fastest total time to identify the hand-signals is ranked higher. The team with the highest total number of points, and lowest total time (for tie-breakers), wins this in-game challenge.

A.2 Modelcard Movenet

MoveNet.SinglePose

Model Details

A convolutional neural network model that runs on RGB images and predicts human joint locations of a single person. The model is designed to be run in the browser using Tensorflow.js or on devices using TF Lite in real-time, targeting **movement/fitness activities**. Two variants are presented:

- **MoveNet.SinglePose.Lightning**: A lower capacity model that can run >50FPS on most modern laptops while achieving good performance.
- **MoveNet.SinglePose.Thunder**: A higher capacity model that performs better prediction quality while still achieving real-time (>30FPS) speed. Naturally, *thunder will lag behind the lightning, but it will pack more of a punch.*

Model Specifications

Model Architecture

MobileNetV2 image feature extractor with Feature Pyramid Network decoder (to stride of 4) followed by CenterNet prediction heads with custom post-processing logic. **Lightning** uses depth multiplier **1.0** while **Thunder** uses depth multiplier **1.75**.

Inputs

A frame of video or an image, represented as an int32 tensor of shape: 192x192x3(**Lightning**) / 256x256x3(**Thunder**). Channels order: RGB with values in [0, 255].

Outputs

A float32 tensor of shape [1, 1, 17, 3].

- The first two channels of the last dimension represents the yx coordinates (normalized to image frame, i.e. range in [0.0, 1.0]) of the 17 keypoints (in the order of: [*nose, left eye, right eye, left ear, right ear, left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist, left hip, right hip, left knee, right knee, left ankle, right ankle*]).
- The third channel of the last dimension represents the prediction confidence scores of each keypoint, also in the range [0.0, 1.0].

Authors

(Equal contributions)

Francois Beletti, Google
Ard Oerlemans, Google
Yu-Hui Chen, Google
Ronny Votel, Google

Intended Use

Primary Intended Uses

- Optimized to be run in the browser environment using Tensorflow.js with WebGL support or on-device with TF Lite.
- Tuned to be robust on **detecting fitness/fast movement with difficult poses and/or motion blur**.
- Most suitable for detecting the pose of a single person who is **3ft ~ 6ft** away from a device's webcam that captures the video stream.
- Focus on detecting the pose of the person who is closest to the image center and ignore the other people who are in the image frame (i.e. background people rejection).
- The model predicts **17 human keypoints** of the full body even when they are occluded. For the keypoints which are outside of the image frame, the model will emit low confidence scores. A confidence threshold (recommended default: 0.3) can be used to filter out unconfident predictions.

Primary Intended Users

- People who build applications (e.g. fitness/physical movement, AR entertainment) that require very **fast inference** and **good quality single-person pose detection** (with background people rejection) on **standard consumer devices** (e.g. laptops, tablets, cell phones).

Out-of-scope Use Cases

- This model is not intended for detecting poses of multiple people in the image.
- Any form of surveillance or identity recognition is explicitly out of scope and not enabled by this technology.
- The model does not store/use/send any information in the input images at inference time.

Evaluation Data

- **COCO Keypoint Dataset Validation Set 2017:** In-the-wild images with diverse scenes, instance sizes, and occlusions. The original validation set contains 5k images (images, annotations) in total. The images which contain a single person are retained to be the evaluation set of this model, in total of 919 images. The dataset is chosen to evaluate the model performance in the general in-the-wild scenario.
- **Active Dataset Evaluation Set**: Images sampled from YouTube fitness, yoga, and dance videos which captures people movements. It contains diverse poses and motion with more motion blur and self-occlusions. The set contains 1161 images with a single person in the frame. This dataset is chosen to evaluate the model performance on the targeted domain, i.e. fitness/human motion.

Training Data

- **COCO Keypoint Dataset Training Set 2017:** In-the-wild images with diverse scenes, instance sizes, and occlusions. The original training set contains 64k images (images, annotations). The images with three or more people were filtered out, resulting in a 28k final training set.
- **Active Dataset Training Set**: Images sampled from YouTube fitness videos which captures people exercising (e.g. HIIT, weight-lifting, etc.), stretching, or dancing. It contains diverse poses and motion with more motion blur and self-occlusions. The set of images with a single person contains 23.5k images.

Factors

Groups

To perform fairness evaluation, we analyze the model performance under different person attributes and categories:

- **Gender**: Male/Female
- **Age**: Young/Middle-age/Old
- **Skin tone**: Medium/Darker/Lighter

Instrumentation

The training dataset images were captured in a real-world environment with different light, noise, and motion. Therefore, the model is robust to the input video streams that are captured through common devices' webcams.

Environments

The model is trained on images with various lighting, noise, motion conditions and with diverse augmentations.

Metrics

- **Keypoint mean average precision (mAP) with Object Keypoint Similarity (OKS)**: this is the standard metric used to evaluate the quality of the predictions of a keypoint model in the COCO competition.
- **Inference Time:** the time spent to run the model inference for a single image measured in milliseconds.

A.3 Code und Datensätze

Das Projekt *Training und Evaluation eines neuronalen Netzes zur Lösung der „Visual Referee Challenge“* und die verwendeten Datensätze befinden sich auf der beiliegenden SD-Karte.

Eigenständigkeitserklärung

Hiermit erkläre ich, Freijdis Jurkat, dass ich die vorliegende Arbeit, eingereicht an der Hochschule für Technik, Wirtschaft und Kultur Leipzig zum erlangen des Bachelortitels, mit dem Thema

Training und Evaluation eines neuronalen Netzes zur Lösung der "Visual Referee Challenge"

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle sinngemäß und wörtlich übernommenen Textstellen aus fremden Quellen wurden kenntlich gemacht.

Leipzig, 15.05.24
Ort, Datum

[redacted]
Unterschrift